VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Juraj Korček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO GENEROVÁNÍ A OVĚŘOVÁNÍ KONFIGURACÍ SÍŤOVÝCH ZAŘÍZENÍ
APPLICATION GENERATING AND VERIFYING CONFIGURATIONS OF NETWORK DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Juraj Korček

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Jan Jeřábek, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Juraj Korček ***ID:*** 187238
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Aplikace pro generování a ověřování konfigurací síťových zařízení

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou síťových zařízení, síťových operačních systémů, hlavních používaných komunikačních protokolů a způsobů konfigurace těchto zařízení. Dále prostudujte problematiku osvědčených postupů konfigurace, zejména s ohledem na bezpečnost fungování zařízení v síti a také problematiku anonymizace těchto konfigurací. Navrhněte systém či aplikaci, která bude umět pro vybranou množinu síťových zařízení vytvářet přednastavené parametry nastavení, které bude možné na dané síťové zařízení aplikovat. Dále musí daná aplikace umět verifikovat správnost existujících konfigurací, upozornit na případné nedostatky a i konfiguraci modifikovat tak, aby splňovala hlavní bezpečnostní a provozní standardy a doporučení. Fungování aplikace ověřte na testovacích vzorcích síťových konfigurací různých zařízení z několika různých sítí a případně i různých výrobců.

DOPORUČENÁ LITERATURA:

[1] Stallings W., Network security essentials: applications and standards. 6th ed. Hoboken: Pearson education, 2017, 445 s. ISBN 978-0-13-452733-8.

[2] McMillan, T., CCNA Security Study Guide: Exam 210-260. 2nd ed. USA: Sybex, 2018, 384 s. ISBN 978-1--1-940993-9.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cieľom tejto diplomovej práce je návrh a následná implementácia programu na nájdenie bezpečnostných a prevádzkových nedostatkov v sieťových zariadeniach, ako aj ich náprava pomocou generovania opravnej konfigurácie. Z dôvodu nedostatočného zabezpečenia a nesprávnej konfigurácie sú mnohé zariadenia v sieti často nevedome vystavené riziku bezpečnostného incidentu. Z tohto dôvodu program porovnáva ich nastavenia s rôznymi štandardmi, odporúčaniami a osvedčenými postupmi a vytvára správu s nálezmi, aby bolo možné tieto nedostatky odstrániť pomocou automaticky vygenerovanej nápravy alebo manuálne, pokiaľ automatická náprav nie je možná. Program využíva na nájdenie problémových nastavení regulárne výrazy, pomocou ktorých hľadá nedostatky vo vyexportovaných konfiguráciách. Jeho implementácia je v jazyku Python a využíva sa aj značkovací jazyk YAML. Vedľajším produktom práce je aj kontrolný zoznam, ktorým sa dá riadiť pri zostavovaní modulov pre podporu ďalších výrobcov, a tým rozšíriť program.

KĽÚČOVÉ SLOVÁ

sieť, zariadenie, smerovač, prepínač, bezpečnosť, overenie, kontrola, audit, generovanie, konfigurácia, nastavenie, python, yaml

ABSTRACT

The aim of this master's thesis is a design and implementation of a program for finding security and operational deficiencies of network devices and afterwards, resolving them by generating corrective configuration. Due to a lack of security and misconfiguration, there are a lot of devices exposed to the risk of a security incident. Therefore, the program compares settings with various standards, recommendations, and best practices and generates a report with findings. Afterwards, deficiencies can be eliminated by automatic resolution or manually if automatic resolving is not possible. The program uses regular expressions to find problem settings in previously exported configurations. Implementation is written in Python, and YAML markup language is used too. Another output of this thesis is a checklist, which can be used for the creation of future modules for support of other network device vendors and thus extend the program.

KEYWORDS

network, device, security, router, switch, verification, check, audit, generation, configuration, setting, python, yaml

KORČEK, Juraj. *Aplikace pro generování a ověřování konfigurací síťových zařízení*. Brno, 2020, 109 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Ing. Jan Jeřábek, PhD.

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Aplikace pro generování a ověřování konfigurací síťových zařízení“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi doc. Ing. Janovi Jeřábkovi Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Kybernetická bezpečnosť je bezpochyby jednou z hlavných tém 21. storočia. Útoky na infraštruktúru a systémy naberajú nielen na frekvencii, ale čo je ešte horšie tak aj na sofistikovanosti. Napriek častému zdôrazňovaniu odborníkov o kladenie čoraz väčšieho dôrazu na bezpečnosť pri návrhu, implementácii a nasadení, sa stále stretávame s fatálnymi dôsledkami, ktoré boli spôsobené nedostatočným venovaním pozornosti bezpečnosti.

Problém nedostatočného zabezpečenia nie je ani tak nevedomosť základných bezpečnostných praktík administrátorov alebo programátorov, ale potreba rýchleho nasadenia systému a infraštruktúry s odložením implementácie bezpečnostných praktík na neskôr. Tieto problémy vznikajú aj pri dodatočnej implementácií nových modulov a pridaní novej infraštruktúry, kedy sa nemení celok, ale pridanie jednej časti môže výrazne ovplyvniť a zmeniť stav bezpečnosti celého systému. Z tohto dôvodu je priam žiadúce disponovať nejakým procesom alebo nástrojom na dodatočné zistenie nedostatkov a ich následnú elimináciu. Veľmi silnou motiváciou by malo byť aj to, že dôsledkom bezpečnostných nedostatkov sú globálne miliardové škody a straty reputácií firiem.

Jednou z hlavných častí infraštruktúry, kde dochádza k významným bezpečnostným incidentom je počítačová sieť, bez ktorej by dnes informačné technológie nevedeli fungovať. Preto sa táto práca bude zaoberať práve ňou, keďže je vstupnou bránou do systémov a jej vyradením alebo zneužitím prichádzajú organizácie o finančné prostriedky, citlivé dáta a dôveru užívateľov.

Výsledkom tejto práce bude aplikácia overujúca nastavenia sieťových zariadení prevažne v lokálnej sieti, ktorá umožňuje zjednať nápravu na základe nájdených nedostatkov. Výhodou oproti existujúcim riešeniam bude otvorenosť kódu a modularita, ktorá umožní rozšírenie aplikácie na sieťové zariadenia rôznych výrobcov. Dôležitým výstupom bude taktiež zoznam bezpečnostných a prevádzkových odporúčaní vychádzajúcich z rôznych štandardov a odporúčaní, ktoré môžu byť v budúcnosti použité ďalšími užívateľmi aplikácie pri zostavovaní modulov pre zariadenia rôznych výrobcov. Vytvorený zoznam odporúčaní bude obsahovať zatriedenie odporúčaní podľa závažnosti, čo súčasné riešenia neponúkajú. Odporúčania ako aj výsledná aplikácia počíta s rozdelením odporúčaní a nálezov aj podľa umiestnenia zariadenia v hierarchickom modeli siete, aby nedochádzalo ku falošne pozitívnym správam. Jednou z kľúčových vlastností je bezplatnosť, keďže podľa zistení takmer polovica útokov smeruje na malé firmy, ktoré bezpečnosť často neriešia z finančnej náročnosti programov na detekciu bezpečnostných nedostatkov.

1 Kybernetická bezpečnosť

S čoraz na väčšou informatizáciou naprieč všetkými odvetviami života, je nutnosťou riešiť aj zabezpečenie systémov, infraštruktúry a dát. Kybernetická bezpečnosť je bez pochýb jednou z najdiskutovanejších tém 21. storočia.

Podľa zistení z roku 2018 [1] takmer polovica útokov smeruje na malé firmy, ktoré bezpečnosť riešia iba minimálne alebo vôbec. Predpokladá sa [1], že pre rok 2019 bude na kybernetickú bezpečnosť minutých 6 miliárd dolárov, naopak škody spôsobené kybernetickými útokmi presiahnu jednu miliardu dolárov a veľmi záškodné útoky typu *Distributed Denial of Service – distribuované odoprenie služby* (DDoS) by mali vzrásť až šesťnásobne.

Vyššie zmienené predpovede len potvrdzujú dôležitosť kybernetickej bezpečnosti pri návrhu, implementácií, nasadzovaní a prevádzke informačných technológií.

1.1 Vybrané pojmy z kybernetickej bezpečnosti

- Informačné aktívum (Asset) – čokoľvek, čo je nutné chrániť, napr. dáta, fyzická informačná infraštruktúra, systémy [2].

- Zraniteľnosť (Vulnerability) – neprítomnosť alebo nedostatočné opatrenia na zabezpečenie. Zraniteľnosť môže byť prítomná hardvéri, softvéri alebo samotnom užívateľovi [2].

- Hrozba (Threat) – vzniká v prípade odhalenia alebo zneužitia zraniteľnosti. Zároveň platí, že hrozbou je aj zraniteľnosť, ktorá doposiaľ nebola neidentifikovaná [2].

- Útočník (Threat agent) – entita, ktorá zneužije zraniteľnosť [2].

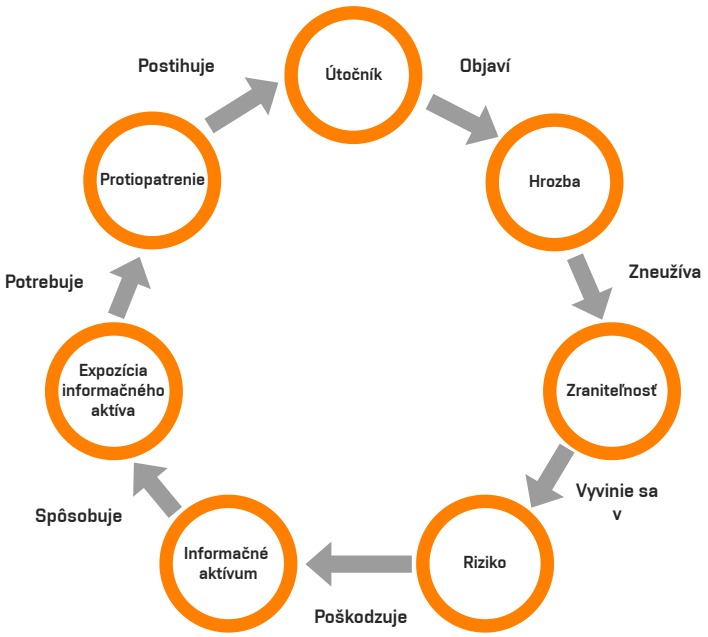
- Riziko (Risk) – pravdepodobnosť, že útočník využije zraniteľnosť, pričom príde k dopadu na systém alebo infraštruktúru [2].

- Útok na bezpečnosť (Security attack/Explotation) – krok, ktorý kompromituje bezpečnosť informačného aktíva [3].

- Bezpečnostný mechanizmus (Security mechanism) – proces, ktorý je navrhnutý na detegovanie, prevenciu a zotavenie z útoku.

- Protiopatrenie (Countermeasure) – ochranné opatrenie, ktoré znižuje riziko [2].

- Expozícia informačného aktíva (Exposure) – dochádza k nej ak je aktívum vystavené stratám nedostatočným alebo neprítomným zabezpečením [2].



Obr. 1.1: Koncept bezpečnosti a vzájomné vzťahy pojmov [2]

Na obrázku 1.1 je možné vidieť vzájomnú interakciu medzi pojmami. Zároveň je nutné si uvedomiť, že takýto cyklus nie je v systéme alebo infraštruktúre jeden a taktiež môže vzniknúť niekoľko paralelných cyklov pričom každý môže mať počiatok v inom uzle. Je dobré myslieť na to, že jednotlivé cykly môžu na seba vplývať, napríklad jedno protiopatrenie môže postihnúť viacero útočníkov využívajúcich rôzne hrozby.

1.2 Ciele sieťovej bezpečnosti

Bezpečnosť počítačovej siete, tak ako aj iných podoblastí kybernetickej bezpečnosti je založená na troch základných princípoch známych ako *confidentiality, integrity, availability* – *dôvernosť, integrita, dostupnosť* (CIA). Bezpečnosť musí pokryť všetky tri aspekty popísané týmto modelom, pričom narušenie čo i len jednej zložky má za následok nesplnenie celkového zabezpečenia [3].

1.2.1 Triáda CIA

Triáda CIA pozostáva z nasledujúcich častí [2]:

- Confidentiality (Dôvernosť) – zabránenie prístupu k dátam alebo informáciám neoprávneným osobám. Na zaistenie tejto požiadavky sa najčastejšie používa šifrovanie, ale aj autentifikácia a autorizácia. Jej strata vedie k neoprávnenému zverejnenie informácií.
- Integrity (Integrita) – dáta alebo informácie sú zabezpečené proti neautorizovanej modifikácií a poškodeniu. Týmto zaisťujem konzistenciu dát pri prenose alebo uchovaní na médiu. Integritu zaisťujeme hašovacími funkciami prípadne za pomoci *Access Control List – zoznam pre riadenie prístupu* (ACL).
- Availability (Dostupnosť) – dáta alebo informácie sú dostupné iba pre určité entity v daný čas a miesto.

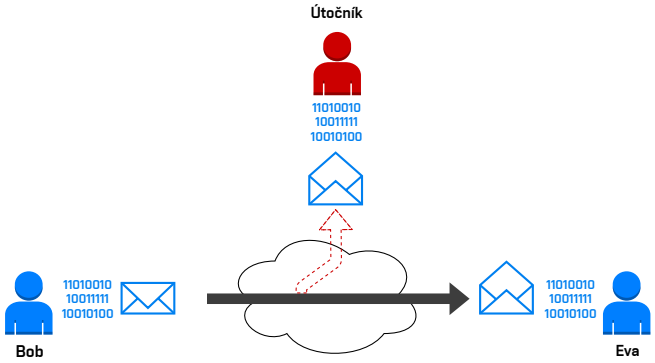
Obr. 1.2: Triáda dôvernosť, integrita a dostupnosť demonštrujúca potrebu všetkých troch prvkov na zaistenie bezpečnosti [3].

Aj keď triáda CIA definuje ciele na zaistenie bezpečnosti, tak niektorí odborníci ju nepovažujú za dostatočnú a zavádzajú ďalšie dve podmienky a pojmy [4]:

- Authencity (Autenticita) – overenie originál, platnosti správy a identity jej pôvodcovi. Najčastejšie sa na zaistenie tejto podmienky využívajú certifikáty.
- Accountability (Sledovateľnosť) – identifikácia prístupu k informáciám a vysledovateľnosť bezpečnostných incidentov v prípade využitia forenznej analýzy. Väčšinou je táto požiadavka zaistená záznamom činnosti v systéme formou logu.

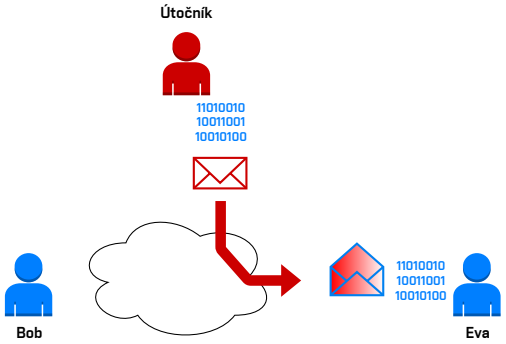
1.3 Pasívne a aktívne útoky

Útoky na bezpečnosť môžu byť rozdelené do dvoch skupín [3]. Jednou skupinou je pasívny útok, kde útočník nepozmeňuje pôvodné dáta a nevplýva na príjemcu týchto dát. Druhou možnosťou je aktívny útok, pri ktorom sú buď pozmenené dáta doručené príjemcovi alebo je obeť nejakým spôsobom ovplyvňovaná, napríklad zasielaním falošných informácií.

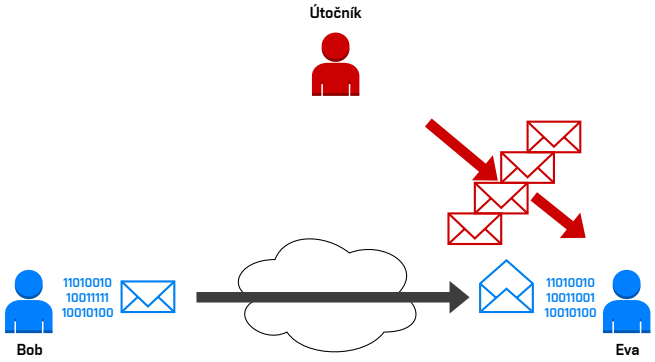


Obr. 1.3: Príklad pasívneho útok, pri ktorom útočník odpočúva komunikáciu medzi dvoma uzlami [4].

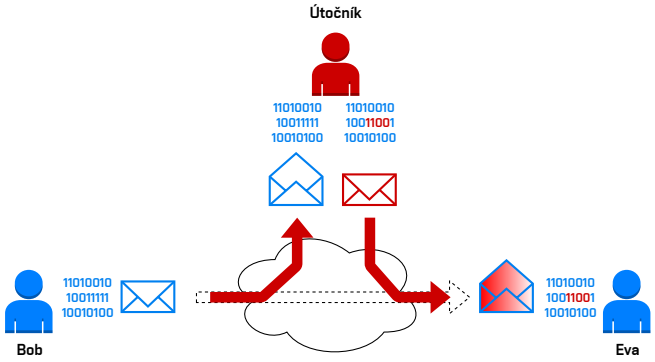
Pri pasívnom útoku, ktorý je znázornený na obrázku 1.3 ide útočníkovi prevažne o zachytenie prenášanej komunikácie, monitorovanie a analýzu prevádzky. Odposluch a zobrazenie obsahu dát je účinné hlavne pri nepoužití šifrovania správ medzi koncovými bodmi alebo aj pri použití slabých šifier, krátkych kľúčov a nedostatočne bezpečných hesiel. Monitorovanie prevádzky, respektíve analýza komunikácie je možná aj pri použití šifrovania, keďže každá komunikácia je charakteristická určitým vzorom. Pasívne útoky je nesmierne obtiažne detegovať nakoľko nemodifikujú dáta pri prenose. Najúčinnejšia obrana je použitie dostatočne silných šifier na zabezpečenie dát. Jeden z pasívnych útokov sa hojne využíva aj pri prevencii v *Intrusion Detection System – systém detekcie narušenia* (IDS) a *Intrusion Prevention System – systém prevencie prienikov* (IPS), kde bez analýzy prevádzky by nebolo možné zabezpečiť sieť. Pasívnymi útokmi sa nespôsobuje škoda na systéme alebo infraštruktúre, ale hrozba spočíva v narušení dôvernosti.



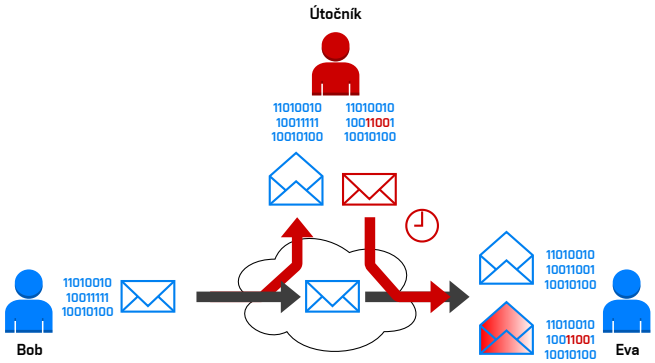
Obr. 1.4: Príklad aktívneho útoku maškarádou, kedy uzol Eva obdrží falošnú správu od útočníka mysliac si, že ide o správu od uzla Bob [4].



Obr. 1.5: Príklad aktívneho útoku DOS, pri ktorom je uzol Eva zahltený nevyžiadanými správami (označené červeno) [4].



Obr. 1.6: Príklad aktívneho útoku modifikáciou správy, pri ktorom je originálna správa presmerovaná cez útočníka, následne pozmenená a prijatá uzlom Eva, ktorý ju považuje za legitímnu [4].



Obr. 1.7: Príklad aktívneho útoku prehratím, pri ktorom príde uzlu Eva legitímna správa (označená modro) a následne po určitom čase aj odchytená správa od útočníka, ktorá je pozmenená (označená červeno) [4].

Aktívne útoky sú sofistikovanejšie ako pasívne, modifikujú dáta alebo vytvárajú falošné, o ktorých prijímateľ predpokladá, že prišli od zdroja, s ktorým pôvodne komunikoval. Hrozby, ktoré môžu týmito útokmi nastať sú strata integrity, teda modifikácia dát a ohrozenie dostupnosti pričom vždy dochádza ku škode na systéme alebo infraštruktúre. Maškaráda je prvým z aktívnych útokov, kde ako je možné vidieť na obrázku 1.4, útočník vytvára falošnú správu, ktorú zasiela obeti a tá sa domnieva, že komunikuje s pôvodným zdrojom, v našom prípade Bobom. Použitím osobných certifikátov na oboch stranách by bolo možné odhaliť, že správa nepochádza od zdroja, ale od útočníka. Príkladom aktívneho útoku je aj útok odoprenia služby 1.5, kde sa vytvárajú falošné dáta generované vysokou frekvenciou (v obrázku značené červenou farbou), za účelom odstaviť systém alebo infraštruktúru, ktorá nezvláda spracovanie toľkých požiadaviek, keďže nebola na takúto záťaž dimenzovaná. Tretím aktívnym útokom 1.6 je modifikácia správy útočníkom pri prechode komunikačným kanálom, ktorý sa realizuje rôznymi technikami podvrhnutia zdroja alebo identity. Komunikácia v tomto prípade prebieha cez útočníka, ktorý tento útok mohol uskutočniť napríklad podvrhnutím smerovania. Posledným útokom je útok prehratím 1.7, čo je útok veľmi podobný predchádzajúcemu, akurát obeť obdrží najprv pôvodnú nepozmenenú správu a následne po určitom čase aj modifikovanú správu od útočníka. Takéto správy môžu byť generované aj ako nežiadúca sieťová prevádzka pri zahltení prvkov alebo pri zlom nastavení smerovania. Citlivé sú najmä transakčné systémy napríklad databáze. Zabrániť tomuto útoku je možné pomocou časových pečiatok a jednoznačných identifikátorov.

2 Bezpečnostný audit

Audit je veľmi dôležitým prvkom správy informačných systémov a infraštruktúry, pretože umožňuje zaistiť bezpečnosť týchto informačných aktív porovnávaním s vytvorenými štandardmi, odporúčaniami a predpismi. Zaoberá sa otázkami čo a ako zabezpečiť, vyhodnocovaním a riadením rizík a následným dokazovaním, že náprava znížila riziko hrozby.

Audit sa skladá z piatich pilierov [5]:

1. Posúdenie
2. Prevencia
3. Detekcia
4. Reakcia
5. Zotavenie

Pri posudzovaní si je potreba klásť otázky či sú prístupové práva dostatočne špecifikované, aká je pravdepodobnosť útoku na zraniteľnosť a podobne. Prevencia nespočíva iba v technológiách ako firewall prípadne IDS a IPS, ale aj v politikách, procesoch a povedomí o probléme. Detekcia a reakcia spolu úzko súvisia a je potrebné skrátiť dobu medzi týmito dvoma bodmi. Bez dôkladnej detekcie nie je možné vykonať reakciu. Mnohé reakcie na detekciu problému sú už rôznymi technológiami implementované automatizovane. Posledný článkom je zotavenie, ktoré je dôležité pri službách vysokej dostupnosti. Výborným príkladom detekcie, reakcie a zotavenia z problému sú protokoly z rodiny *First Hop Redundancy Protocol* (FHRP).

Proces auditu pozostáva z niekoľkých fáz: [5]

1. Plánovanie – stanovenie cieľov a predmetu auditu. Definuje sa rozsah, teda čo všetko je v pláne auditom pokryť.
2. Výskum – vytváranie auditného plánu na základe štandardov a odporúčaní a špeciálnych expertíz. Kontaktujú sa tiež dotknuté strany, ktoré nám môžu byť nápomocné pri plnení cieľov.
3. Zbieranie dát – vyžiadanie potrebných podkladov a dát na vykonanie auditu, zozbieranie dôkazov. V tejto fáze sa tiež vyberajú rôzne softvérové nástroje na vykonanie auditu a vytvorí sa kontrolný zoznam na základe auditného plánu a zozbieraných dôkazov.
4. Analýza dát – posúdenie všetkých dôkazových dát pomocou kontrolného zoznamu a softvéru na podporu auditu. Na základe nájdených nedostatkov sa vytvoria odporúčania, ktoré by mali znížiť riziká hrozieb.
5. Vytváranie správy – súpis nájdených nedostatkov, možných riešení na zníženie rizík do auditnej správy a prezentácia tejto správy dotknutým stranám.

6. Aplikácia opatrení – nasadenie a použitie protiopatrení prezentovaných alebo vyplývajúcich z auditnej správy. Následne sa môže vykonať monitorovanie a hlásenie o úspešnosti zmien.

Typy auditov podľa zistení, hĺbky a rozsahu auditu:

- Bezpečnostná kontrola – je najzákladnejšia forma analýzy bezpečnosti, na základe ktorej sa následne formujú ďalšie aktivity na zaistenie bezpečnosti. Do tejto kategórie spadajú automatizované nástroje na skenovanie zraniteľností a penetračné nástroje, ktoré generujú zoznam potenciálnych zraniteľností. Výsledky je potrebné podrobnejšie preskúmať a stanoviť si, ako sa k ním zachovať. Patria sem nástroje ako napríklad Nmap, Nessus a podobne. Za bezpečnostnú kontrolu možno považovať preskúmanie politík alebo architektúry daného systému a infraštruktúry. Dá sa povedať, že ide o akýsi rýchly náhľad na bezpečnosť, ktorého výstupom je poznanie a identifikovanie problému.
- Hodnotenie bezpečnosti – je ďalším stupňom, pričom ide o podrobnejší pohľad na problém z profesionálnejšieho hľadiska. Kvalifikuje sa riziko k jednotlivým zisteniam a stanovuje sa relevantnosť a kritickosť týchto zistení na konkrétnu organizáciu a prípad použitia.
- Bezpečnostný audit – je štandardizovanou a najdôkladnejšou formou posúdenia bezpečnosti. Bezpečnosť sa porovnáva so štandardmi alebo benchmarkmi, v niektorých prípadoch aj s predpismi dohliadajúcich orgánov. Výsledkom je posúdenie, na koľko je organizácia alebo skúmaný objekt v zhode s porovnávaným štandardom. Typickým príkladom štandardov sú ISO27001, ITIL, COBIT.

2.1 Manažment rizík

Manažment rizík je proces pozostávajúci z analýzy rizík a riadenia rizík [2]. Dôležitým faktom je, že riziko nie je možné eliminovať, ale ho iba znížiť.

Pri analýze rizík zisťujeme, aké riziká existujú, ako medzi sebou súvisia a aké škody môžu spôsobiť. Analýza rizík môže byť vykonávaná kvalitatívne a kvantitatívne.

Štandard NIST SP 800-30 [6] definuje nasledujúce kroky pri analýze rizík:

1. Identifikácia informačných aktív a ich význam
2. Identifikácia hrozieb
3. Identifikácia zraniteľností
4. Analýza riadenia a kontroly
5. Zistenie pravdepodobnosti
6. Identifikovanie dopadu
7. Definovanie rizika ako súčinu pravdepodobnosti a dopadu
8. Odporúčanie na zavedenie riadenia a kontroly na zníženie rizika
9. Zdokumentovanie výsledkov

Riadenie rizík má za úlohu minimalizáciu potenciálnych škôd odhalených pri analýze rizík s ohľadom na vyváženie nákladov na riadenie rizika.

Prístupy k nájdenému riziku [3][2][5]:

- Vyhnutie sa riziku – je uplatnené ak prítomnosť a funkčnosť informačného aktíva nestojí za podstúpenie rizika, a teda toto aktívum vôbec nepoužijeme. Napríklad vypnutie menej bezpečných a nevyužívaných sieťových služieb.
- Zníženie – aplikovanie protiopatrenia na odstránenie hrozby alebo zraniteľnosti, prípadne zníženie pravdepodobnosti rizika. Nikdy nie je však možné riziko eliminovať. Príkladom môže byť obmedzenie prístupu k sieťovému prvku.
- Akceptovanie – v prípade neexistujúceho protiopatrenia alebo veľmi nízkeho rizika. Častokrát ide o bezpečnostnú chybu softvéru v službe, ktorú využívame a nie je možné ju vypnúť ani aplikovať protiopatrenie.
- Presun – riziko je možné presunúť na inú organizáciu, napr. poistenie v prípade škody spôsobenej nedostatočným zabezpečením.
- Ignorácia – úplné vypustenie faktu, že dochádza k riziku, tento prístup sa považuje za iracionálny.

Na ohodnotenie rizika slúžia rôzne systémy hodnotenia, jedným z nich je *Common Vulnerability Scoring System* (CVSS), ktorý definuje riziká podľa definovaných metrík na základe dosiahnutého skóre do nasledujúcich tried:

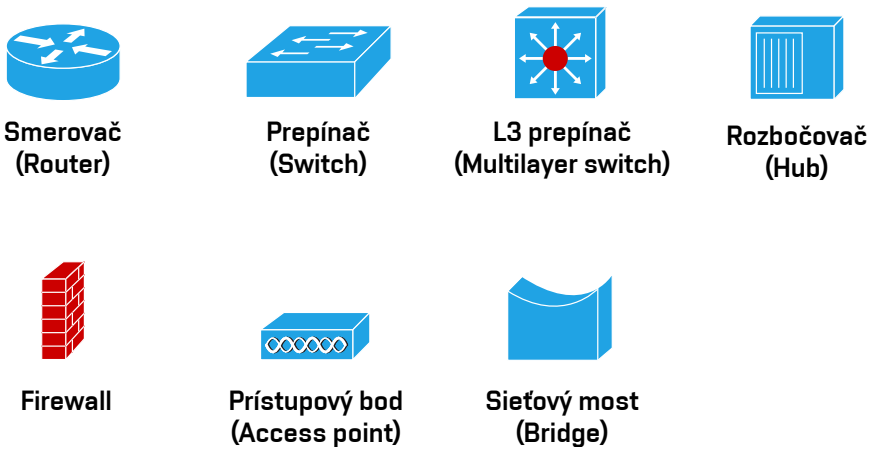
- 0: No issue
- 0,1 – 3,9: Low
- 4,0 – 6,9: Medium
- 7,0 – 8,9: High
- 9,0 – 10,0: Critical

3 Prevádzka a bezpečnosť sietí

Prevádzka sieťových zariadení je proces nielen o monitorovaní incidentov, zabezpečovaní konzistencie a konvergencie siete, ale aj o aktualizáciách softvéru a hardvéru, aplikovaní bezpečnostných zásad a politík. Táto kapitola preto opisuje jednotlivé aspekty s ktorými sa pri prevádzke siete môžeme stretnúť.

3.1 Sieťové prvky

Medzi základné stavebné piliere sietí, bez ktorých nie je možná komunikácia koncových staníc patria smerovače (router) a prepínače (switch). Mimo týchto dvoch základných zariadení sa v *Local Area Network* (LAN) sieťach často vyskytujú prístupové body (access point), firewally, sieťové mosty (bridge) a v dnes už ojedinelých prípadoch ešte aj rozbočovače (hub). V súčasnosti však jedno zariadenie môže kombinovať funkcie zariadení, ktoré majú podľa modelov TCP/IP alebo ISO/OSI na starosti inú vrstvu modelu. Preto sa dnes hlavne z finančných dôvodov používajú takzvané L3 prepínače, ktoré s určitými obmedzeniami vedia nahradiť nákladné smerovače. Taktiež smerovače ako aj L3 prepínače umožňujú filtrovanie paketov, takže vedia čiastočne zastať aj základné funkcie firewallu. Značky najpoužívanejších sieťových zariadení sú vyobrazené na obrázku 3.1 a budú používané v nasledujúcich kapitolách.



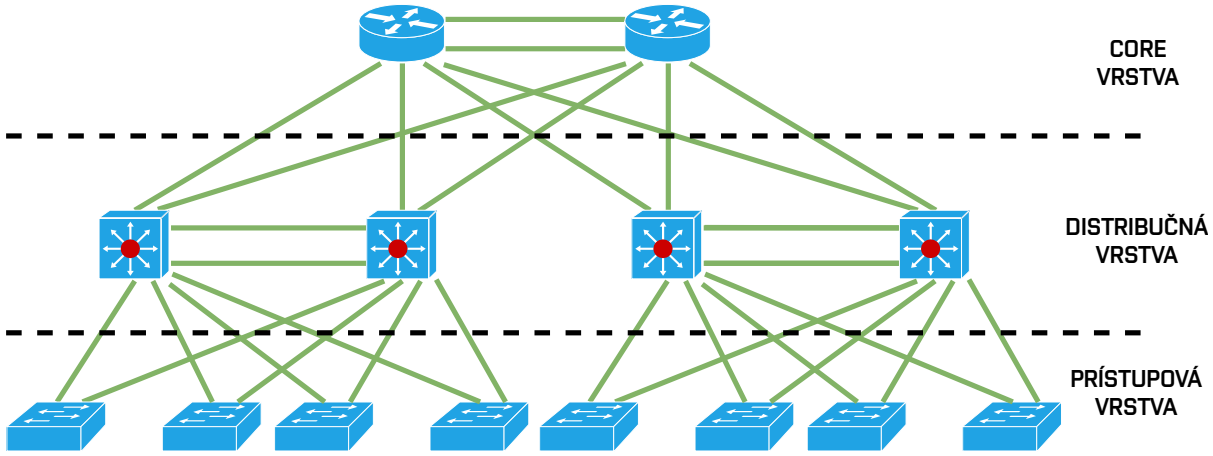
Obr. 3.1: Typy sieťových zariadení v lokálnych sieťach

3.2 Hierarchický model sietí

S postupným nárastom sieťových zariadení a komplexnosti siete dochádza v sieťach bez hierarchie k mnohým problémom ako veľké broadcast domény, vysoká cena za port, vysoké zaťaženia zariadení, neprítomnosť redundancie. Preto sa zaviedol hierarchický model siete, ktorý rieši problémy veľkosti a rozsahu broadcast a kolíznych domén, umožňuje efektívne prideľovanie *Internet Protocol* (IP) / *Internet Protocol version 6* (IPv6) adries a oddeľuje zariadenia pracujúce na jednotlivých vrstvách ISO/OSI.

Siete sú spravidla delené do 3 vrstiev s definovanými funkciami [7]:

- Core – tvorí vysokorýchlostnú chrbticu siete, agreguje dáta z distribučnej vrstvy a mala by byť redundantná. Nároky na rýchlosť portov a výkon zariadenia sú obzvlášť vysoké, a preto sa využívajú prevažne smerovače, ale taktiež ako v distribučnej vrstve dnes už aj L3 prepínače.
- Distribučná (Distribution) – agreguje dáta z prístupovej vrstvy, vytvára a oddeľuje broadcast domény, riadi smerovanie medzi *Virtual LAN* (VLAN) a filtrovanie paketov. Táto vrstva kvôli zabezpečeniu dostupnosti využíva agregovanie a redundanciu liniek. Typicky sa skladá zo smerovačov, no v dnešnej dobe hlavne z L3 prepínačov, keďže tie nie sú finančne také náročné.
- Prístupová (Access) – vstupný bod do siete, ktorý riadi prístup a politiku pre koncové zariadenia, segmentuje sieť, vytvára a separuje kolízne domény. V neposlednej rade zariaďujú prístup k distribučnej vrstve. Je tvorená zariadeniami ako prepínač, rozbočovač alebo prístupový bod.



Obr. 3.2: Hierarchické rozdelenie siete na vrstvy

V menších sieťach prevažne malých firiem sa využíva zlučovanie vrstiev nazývaných ako collapsed core, ktoré zlučujú distribučnú a core vrstvu, prípadne zlučujú všetky tri vrstvy dokopy.

Cieľom hierarchického modelu a dobre navrhnutej siete je dosiahnutie nasledujúcich vlastností:

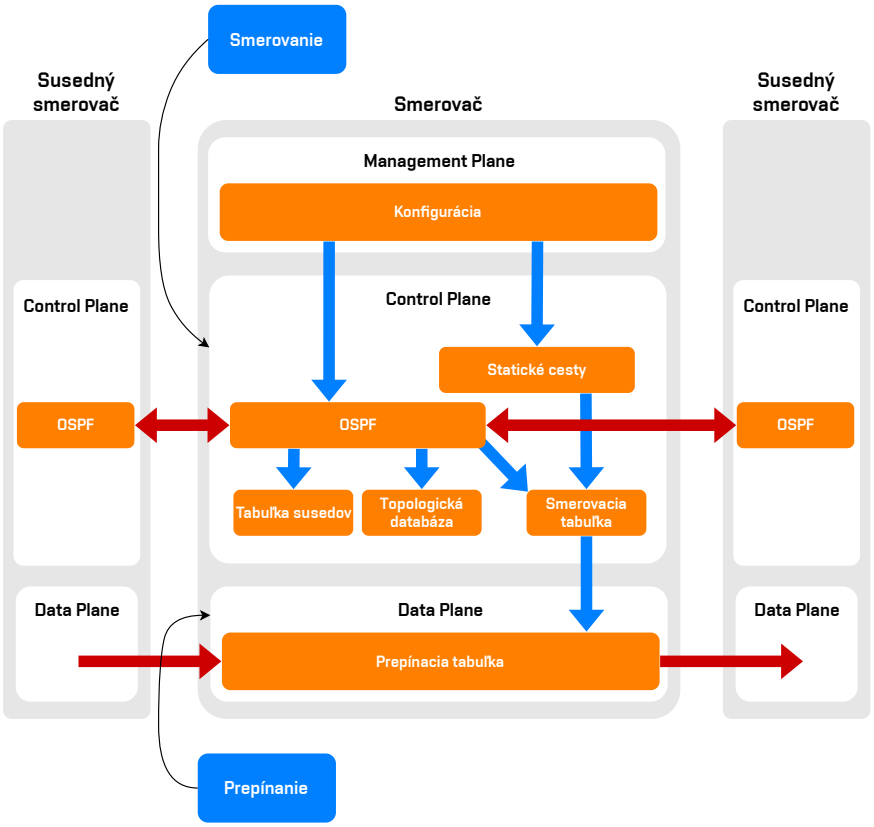
- Škálovateľnosť – jednoduché a bezproblémové pridanie zariadenia pri raste a rozširovaní siete.
- Redundancia – zabezpečenie vysokej dostupnosti viacnásobnými linkami medzi zariadeniami a zálohovanie samotných zariadení ich redundanciou.
- Výkonnosť – agregovanie liniek a výber dostatočne výkonných zariadení.
- Bezpečnosť – zabezpečenie siete na viacerých úrovniach ako napríklad portoch, oddelením segmentov pomocou VLAN, riadením prístupu, šifrovaním a pod.
- Manažovateľnosť – vytvorenie šablón, definovaných štandardov a pravidiel na zaistenie konzistentnosti konfigurácií zariadení na jednoduchšie odhaľovanie chýb.
- Udržovateľnosť – schopnosť systému prechádzať zmenami komponentov, služieb a vlastností.

3.3 Funkčné roviny sieťových prvkov

Sieťové prvky sú zodpovedné nielen za preposielanie dát medzi koncovými stanicami, ale aj za mnohé riadiace dáta medzi sebou, bez ktorých by sieť nebola funkčná. Preto sa jednotlivé protokoly a služby rozdeľujú troch rovín (plane), a to management, control a data plane. Tieto pojmy sa využívajú vo väčšej miere v softvérovo definovaných sieťach, no sú platné aj v klasickej koncepcii.

Rovina management je zodpovedná za konfiguráciu a správu zariadení a riadenie prístupu ku konfiguráciám. Typickými príkladmi protokolov pracujúcich v tejto rovine sú *Simple Network Management Protocol* (SNMP), *Authentication Authorization Accounting* (AAA), Syslog, *Secure Shel* (SSH) a mnohé ďalšie [8]. Druhá rovina, control plane má na starosti prevažne riadenie siete a smerovanie. Zaoberá sa otázkou kadiaľ budú pakety smerované a prenáša riadiace a signalizačné informácie pre protokoly ako napríklad, *Open Shortest Path First* (OSPF), Spanning tree, FHRP [8]. Poslednou rovina je data plane nazývaná často aj forwarding plane, ktorá prepína pakety na daný port na základe rozhodnutia z control plane. Táto časť sieťových prvkov musí byť veľmi rýchla, aby zaistila nízku odozvu a dostatočne vysoké prenosové rýchlosti. Nižšie uvedený obrázok 3.3 reflektuje tok dát z jednej roviny do druhej a tiež medzi dvoma susednými zariadeniami. Rovina management plane je zodpovedná za konfiguráciu zariadenia a nastavuje rovinu control plane,

v tomto prípade smerovanie zariadení. Po výmene informácií so susednými smerovačmi sa vytvoria príslušné tabuľky a nakoniec smerovacia tabuľka, ktorá sa využíva pri rozhodovaní prepínania paketov v rovine data plane.



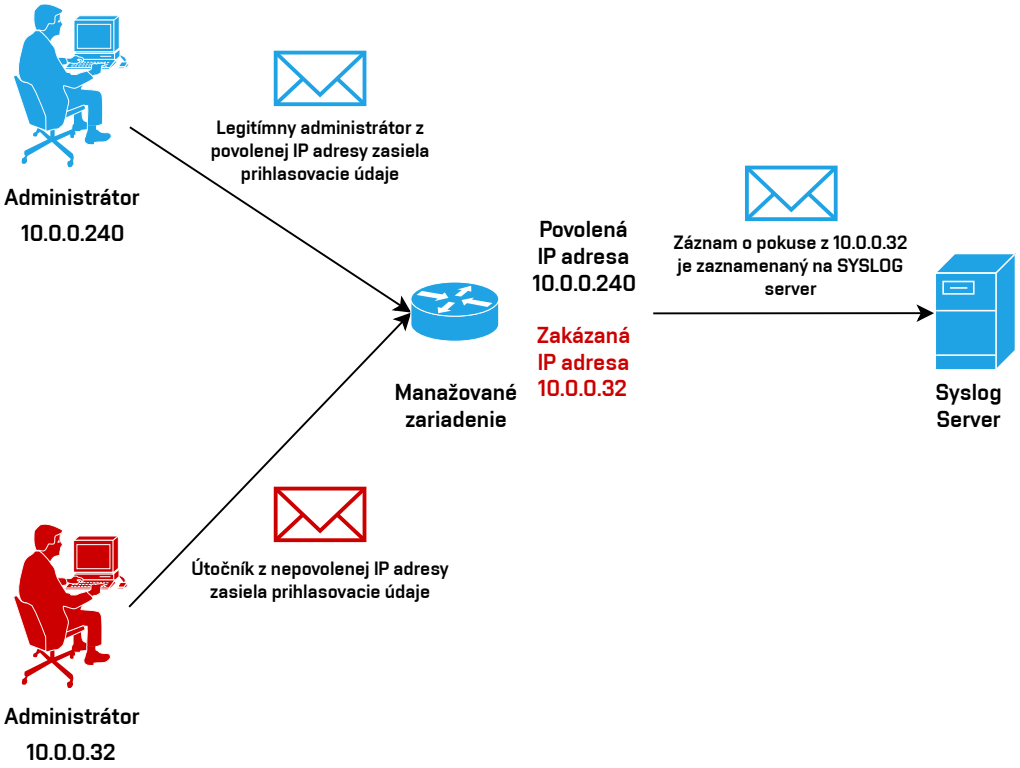
Obr. 3.3: Rozdelenie rovín v smerovači, tok informácií v jeho vnútri a medzi susednými smerovačmi [9].

3.4 Prevádzkové a bezpečnostné postupy

3.4.1 Riadenie a zneužitie prístupu k manažmentu zariadenia

Kritickým miestom často absentujúcim zabezpečenie je prístup ku konfiguráciám zariadení. Typickým príkladom je využitie protokolu Telnet, ktorý nešifruje spojenie a je teda ľahko odpočúvateľný. Preto sa odporúča využívať protokol SSH, naviac je dobré využiť bezpečnú verziu 2 s rozumnou dĺžkou kľúča odpovedajúcou aktuálnym odporúčaniam [10][11]. Jedným z opatrení na zabezpečenie SSH prístupu je zmena portu, na ktorom obvykle počúva z dôvodu, že útočník skúša periodicky útoky hrubou silou na *Transmission Control Protocol* (TCP) port 22. Alternatívou na zabezpečenie SSH prístupu môže byť port knocking, ktorý na základe autorizácie dynamicky povolí záznam v ACL k portu, na ktorom počúva SSH.

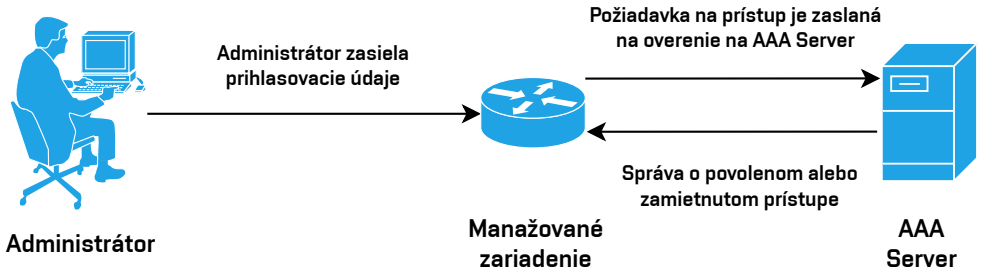
Pri pokusoch o prihlásenie sa často využíva hádanie hesiel, preto je dobré určiť maximálny počet neúspešných pokusov a definovať čas, po ktorý bude prihlásenie zablokované. Riadenie prístupu k manažmentu zariadení by malo byť výhradne z obmedzeného rozsahu staníc administrátorov, na to poslúžia obmedzenia pomocou ACL, aby neprišlo k nechcenému prihláseniu alebo útoku (D)DoS z nechcených klientský staníc. Je tiež dobré zaznamenávať neúspešné ale aj úspešné prihlásenia do manažmentu zariadenia. V prípade konfigurácie viacerými administrátormi naraz môže vzniknúť konflikt, a preto je dobré zabezpečiť, aby v jednom okamihu mohol zmeny vykonávať iba jeden administrátor. Problémom môžu byť aj dlhé aktívne pripojenie k manažmentu zariadenia, ktoré môže byť zneužité pri odblokovanom počítači administrátora. Pri pokuse o prihlásenie alebo zmene nastavení je dobré informovať oznámením alebo správou potenciálneho útočníka s následkami, ktoré mu hrozia v prípade zneužitia zariadenia [10].



Obr. 3.4: Prihlasovanie k manažmentu zariadenia z povolených IP adries a logovanie pokusov z nepovolených IP adries.

Ďalšou obranou proti nechcenému prístupu na sieťové prvky je vytvorenie lokálnych účtov, ktoré budú použité na prihlasovanie a pri zmenách konfigurácie. Bez znalosti kombinácií mena a hesla by nemalo byť umožnené zmeniť nastavenia zariadenia.

Najlepším riešením pre riadenie prístupu k manažmentu zariadenia a účtovaniu sú protokoly spadajúce do skupiny AAA. Patria sem protokoly `Radius`, `TACACS+` alebo `Kerberos`. Tieto protokoly umožňujú okrem riadenia prihlásení administrátorov taktiež špecifikovať príkazy konfigurácie, ktoré budú jednotlivcom povolené a tiež zaznamenávať zmeny jednotlivých administrátorov v konfigurácií, ktoré učinili a naviac aj kedy boli na zariadení prihlásení. Zároveň je treba určiť aj mechanizmus prihlásenia pri výpadku autentifikačného serveru, teda napríklad nejaké záložné lokálne konto.



Obr. 3.5: Overenie prihlásenia k manažmentu zariadenia pomocou AAA serveru.

3.4.2 Filtrovanie prevádzky

Filtrovanie prevádzky môže prebiehať pomocou ACL, teda súborom pravidiel na vrstve L3 a L4, ktoré povoľujú alebo zakazujú komunikáciu. Jedným z dobrých praktík je filtrovať pakety so zdrojovou adresou privátnych alebo špeciálnych adries v smere do vnútornej siete z internetu [5]. Rozsahy týchto IP adries sú definované v RFC 6890 [12] a v RFC 8190 [13]. Administrátori často zabúdajú pri implementácií IPv6, že aj tento protokol má špeciálne adresy, ktoré nemôžu byť zdrojové. Špeciálne IPv6 adresy sú definované v RFC 5156 [14] a aj vo vyššie spomenutých RFC. V prípade použitia týchto špeciálnych adries ako zdrojových sa jedná o útok typu IP Spoofing.

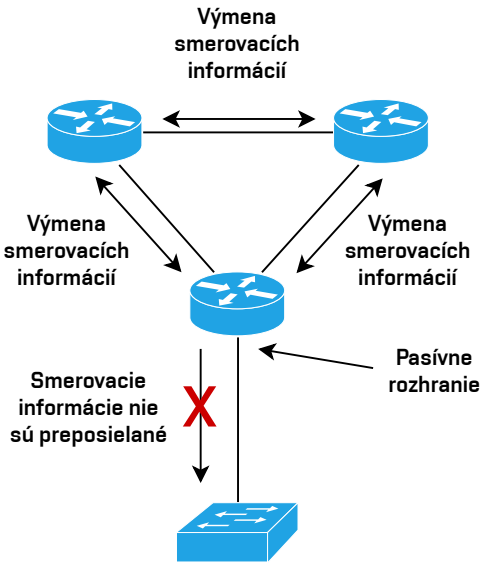
Protokol IPv4 umožňuje pomocou poľa Options vynútenie smerovania na základe zdrojovej adresy a definovanie cesty paketu, ale aj mnohé iné funkcionality. Toto pole nie je však používané a odporúča sa zahadzovať pakety obsahujúce pole Options [8]. Jedným z dôvodov je preťaženie smerovača, keďže každý paket sa musí spracovávať

v procesore a nemôže byť akcelerovaný v špeciálnych obvodoch a tým urýchlené jeho preposlanie.

Protokol IPv6 nemá pole Options, ale tzv. Next Header. Problémom s rozšírenou hlavičkou sa zaoberá v článkoch Martin Grégr a Tomáš Podermański [15][16]. V tomto poli môže byť definovaný protokol vyššej vrstvy, napríklad TCP, ale aj rozšírená hlavička, preto má smerovač problém často rozpoznať, ktoré z týchto dvoch sa nachádza v políčku a ako sa má zachovať, teda či ide o rozšírenú hlavičku alebo nový protokol. To čo nastane s paketom a ako sa zariadenie zachová, závisí od implementácie softvéru v smerovači. Ten sa môže reštartovať, hlavičku preskočiť, zahodiť paket alebo na základe nesprávneho spracovania preskočiť pravidlo zahodenia paketu. Z týchto dôvodov a hlavne preskočením filtrácie sú rozširujúce hlavičky nebezpečné. Administrátor sa môže k ním stavať viacerými spôsobmi, a to zahodením každého paketu s rozširujúcou hlavičkou, zahodenie paketu s neznámou hlavičkou alebo ignorovanie tohto problému. Keďže niektoré rozširujúce hlavičky sú potrebné a používané, tak je dobrou zásadou odfiltrovať práve tie, ktoré zariadenie nevie rozpoznať. V súvislosti s rozšírenými hlavičkami sa zneužíva aj fragmentácia, a to takým spôsobom, že paket sa rozdelí na malé časti a rozšírená hlavička je až v poslednom fragmentovanom pakete. Predpokladá sa, že zariadenie si nevie znovu poskladať a spracovať takto zreťazenú hlavičku a preto dôjde k obídeniu filtrovacích pravidiel. Touto problematikou sa zaoberá RFC 7112 [17], ktoré definuje, aby rozšírená hlavička bola už v prvom pakete a tým bolo možné rozpoznať o akú rozšírenú hlavičku ide. Plošné zakázanie rozširujúcich hlavičiek nie je dobré, keďže sa využíva aj pre IPSec.

3.4.3 Smerovacie protokoly

Používaním dynamických smerovacích protokolov prichádza sieť o určitú časť bezpečnosti, a to vysielaním informácií o pripojených a naučených sieťach a cestách, ktoré môže útočník odchytávať. K tomu sa ešte môže pridať vloženie falošnej informácie a teda zaistenie smerovania cez útočníka. Našťastie obrana proti týmto útokom existuje, aj keď nie je vždy ideálna. V prípade vloženia informácie alebo cesty do správ, ktoré si vymieňajú dynamické smerovacie protokoly je možnou obranou autentifikácia správ poslaných medzi smerovačmi [2][8][10]. Pri zasielaní sa používa hash hesla, a to sa pri prijatí druhým smerovačom porovná s vopred definovaným. Na obrázku 3.6 je vidno, že informácie dynamického smerovacieho protokolu sú zastavené na pasívnom rozhraní [18], a teda užívatelia alebo útočník nemá možnosť sa tieto údaje dozvedieť.



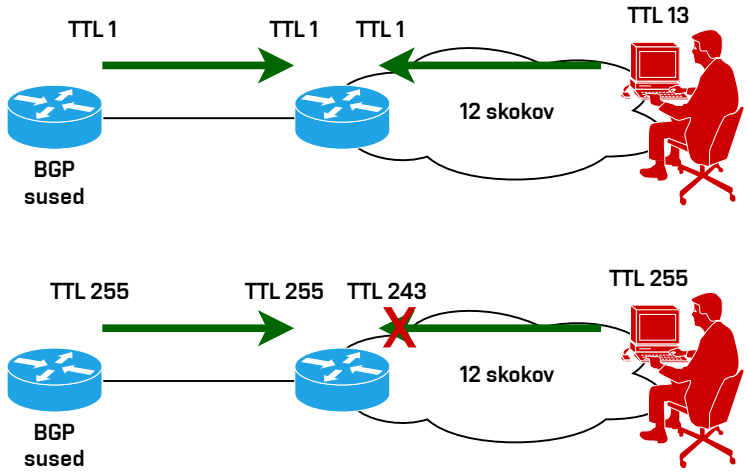
Obr. 3.6: Blokovanie správ dynamického smerovacieho protokolu na pasívne rozhranie.

Bezpečnostnou hrozbou môže byť aj smerovanie na základe zdrojové adresy, pri ktorej si zdroj určí cestu, ktorou bude paket prechádzať namiesto aby túto skutočnosť prenechal na rozhodnutí smerovačov po ceste k cieľu [10]. Táto funkcia využíva pole Options, ktoré býva však často ignorované prípadne pakety s týmto polom zahadzované z bezpečnostných dôvodov. Source Routing pozná dva módy, a to Strict a Loose, v prvom prípade musí paket prejsť všetkými definovanými bodmi a žiadnym iným. Naopak mód Loose definuje uzly, ktoré je potreba navštíviť, no zároveň môžu byť navštívené aj iné uzly po ceste.

Podvrhnutie IP adresy, tzv. IP Spoofing je jedným z útokov, ktorým musia smerovače čeliť. Dá sa mu zbrániť pomocou *Unicast Reverse Path Forwarding* (uRPF) [5], ktorý funguje buď v Strict alebo Loose móde a zisťuje prítomnosť zdrojovej zdrojovej IP adresy. Ako už názov napovedá, tak mód Strict je prísnejší, pretože zahadzuje pakety, ktorej zdrojová adresa sa nenachádza v smerovacej tabuľke a zároveň testuje či zdrojová adresa je dosiahnuteľná cez rozhranie, na ktorom bol paket prijatý. Tento mód je preto nevhodný pri asymetrickom smerovaní. Mód Loose testuje prítomnosť zdrojovej adresy iba v smerovacej tabuľke.

Protokol *Border Gateway Protocol* (BGP) okrem autentifikácie obsahuje aj ďalšiu ochranu, a to *Time To Live* (TTL) security [19]. Pri tomto prístupe sa porovnáva hodnota poľa TTL v pakete, ktorý dorazí do smerovača a známy počet skokov, ktorý sa nakonfiguruje medzi našim smerovačom a zdrojom. Mohlo by sa zdať, že priamo

pripojené siete, teda susedné autonómne systémy týmto problémom netrpia, no pole TTL sa dá zmeniť tak, aby po príchode na smerovač obete malo toto pole hodnotu 1, čo je predvolené TTL, ktoré zasiela BGP, viď obrázok 3.7. Z tohto dôvodu sa používa obrátená forma kontroly, a to testovanie voči maximálnej hodnote TTL, čo je hodnota 255. To znamená, že všetky pakety od priamo pripojených BGP susedov budú mať po príchode na náš smerovač hodnotu TTL 255, tie ktoré to nebudú splňovať sú brané ako nelegitímne pakety, viď 3.7. Treba dodať, že v prípade že smerovače nie sú priamo pripojené, tak je možné použiť aj definovanie vzdialenosti medzi smerovačmi, teda počet skokov, aby susedný smerovač dostal BGP správu. Bezpečnejšie je však použiť TTL security, tak že sa od čísla 255 odpočíta počet skokov medzi dvoma autonómnymi systémami a voči tejto hodnote sa bude robiť kontrola.



Obr. 3.7: Porovnanie prístupov TTL security, kde sa v prvom prípade používa implicitná hodnota 1 na porovnanie TTL a v druhom prípade maximálna hodnota TTL [19].

3.4.4 Identifikácia zariadení, pravidiel a nastavení

K lepšej identifikácií je dobrým pravidlom každé sieťové zariadenie vhodne pomenovať kombináciou typu zariadenia, vrstvy hierarchického modelu, na ktorej operuje a prípadne umiestnenia v racku, napríklad `sw01-dist-rack1`. V súvislosti s týmto nastavením sa často nastavuje aj doména, v ktorej je zariadenie umiestnené. Tieto dve prerekvizity potom umožňujú aj vzdialenú správu zariadenia a prístup cez SSH [10].

Dôležitým prvkom sú komentáre k pravidlám v ACL, ktoré by mali nielen identifi-

kovať, čo presne dané pravidlo povoľuje a zakazuje, ale aj identifikovať požiadavok, na základe ktorého bolo pravidlo vytvorené. Komentáre s popisom je dobré pridávať aj na rozhrania sieťových zariadení, napríklad s popisom, k akému zariadeniu dané rozhranie vedie. Posledným, ale nemenej dôležitým je pomenovanie VLAN pre ich ľahšiu identifikáciu.

3.4.5 Šifrovanie hesiel

Pri úniku konfigurácií môže dôjsť k odhaleniu hesiel uložených v nich, preto by mali byť v konfiguračnom súbore všetky heslá zahašované pomocou čo najpokročilejších hašovacích funkcií, ktoré dané zariadenie podporuje [10].

3.4.6 Logovanie

Záznam činnosti zariadenia patrí k základným prvkom monitorovania sieťovej infraštruktúry spolu s notifikovaním o vzniknutých incidentoch. Na tieto účely sa používajú prevažne dva protokoly, a to SNMP a Syslog.

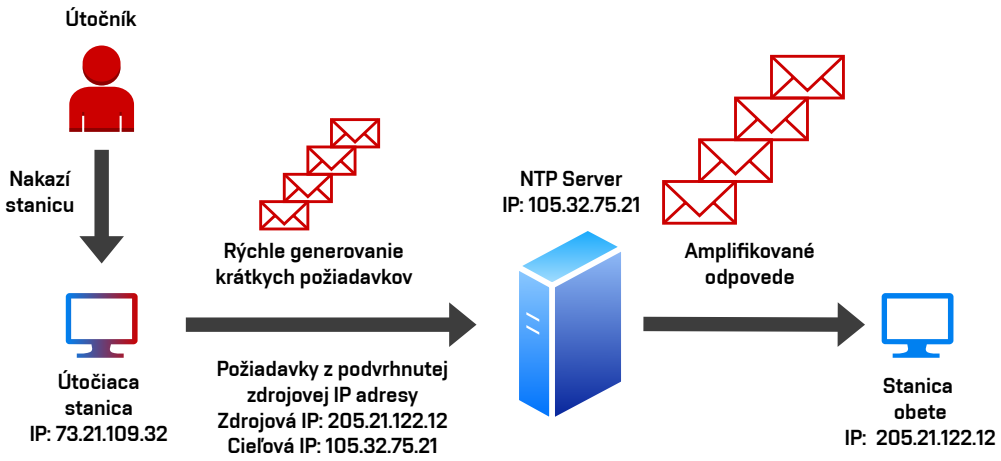
Protokol SNMP využíva buď štandardizovanú databázu MIB, alebo rozšírenú daným výrobcom zariadenia. Jednotlivé monitorované objekty sú v tejto databáze organizované v stromovej štruktúre. V súčasnosti sa využívajú prevažne SNMP verzie 2c a 3. Je vysoko odporúčané využívať verziu 3, ktorá zabezpečuje ako integritu, tak aj dôvernosť a autentifikáciu [10][20]. Protokol SNMP umožňuje pomocou jednotlivých objektov meniť nastavenia zariadení, túto funkciu je však dobré vypnúť a povoliť iba čítanie objektov z presne definovaných IP adries pomocou ACL a asynchrónne správy TRAP.

Druhou možnosťou monitorovania a notifikovania o incidentoch je protokol Syslog. Typicky sa nastavuje Syslog server, ktorý zbiera správy z viacerých zariadení, ktoré môžu byť následne spracovávané špeciálnymi programami a vizualizované v dohľadových centrách. Protokol Syslog pozná 8 úrovní dôležitosti (severity), pričom čím je číslo dôležitosti nižšie, tým ide o závažnejší problém. Pri výpadku Syslog serveru je nutné záznamy ponechať na zariadení a preto mať dostatočné množstvo pamäte [8][21]. V niektorých prípadoch môžu zariadenia vygenerovať väčšie množstvo správ, ktoré majú rovnaký čas a z tohto dôvodu by mali mať jednotlivé správy s rovnakým časom vzniku jednoznačné sekvenčné číslo, aby bolo možné zistiť postupnosť, v akom vznikli incidenty, napríklad zmeny v susedstvách dynamických smerovacích protokoloch.

Veľa útokov mieri práve na protokol SNMP, a preto ho mnohí odporúčajú vypínať [10], na druhej strane protokol Syslog nezabezpečuje žiadnu časť z triády CIA.

3.4.7 Synchronizácia času

Správny a aktuálny čas je dôležitý hlavne pre správne fungovanie certifikátov a protokolu Syslog. V prípade protokolu Syslog zabezpečuje jednoznačnú identifikáciu incidentu v správnom čase a teda lepšiu dohľadatelnosť a určenie vzniku problému. Keďže tento protokol využíva *User Datagram Protocol* (UDP), tak je náchylný na DDoS reflektívne amplifikačné útoky. Tento typ útoku využíva krátku správu zasielanú na NTP server s podvrhnutou zdrojovou IP adresou (IP Spoofing), na ktorú budú zasielané odpovede s oveľa väčšou veľkosťou ako boli pôvodné správy na NTP server. Bohužiaľ obrana proti tomuto typu útoku je veľmi ťažká, poskytovatelia pripojenia k internetu sa s týmto neduhom väčšinou dokážu popasovať [22], v lokálnych sieťach môže pomôcť IP Snooping. Podľa Network Time Foundation [23], aktuálna verzia protokolu 4 nepodporuje šifrovanie správ, no poskytuje akú-takú bezpečnosť pre koncových NTP klientov pomocou MD5, a to autentifikáciu NTP serveru a kontrolu integrity. Naviac protokol nepodporuje žiadnu distribúciu kľúčov. Protokol NTP verzie 4 podporuje aj asymetrickú kryptografiu pomocou Autokey, no podpora tohto riešenia je veľmi slabá [23], jedným z dôvodov je aj náročnosť výpočtov. NTP podporuje sťahovanie správ aj od klientov, toto je výhodné pri prerušení linky ku NTP serveru a na krížovú kontrolu času. Dôležitým nastavením je aj správne časové pásmo, ktoré je dobré zjednotiť naprieč všetkými spravovanými zariadeniami. V prípade roztrúsenia zariadení cez viacero časových pásiem je dobré využívať univerzálny čas UTC. Okrem protokolu NTP existuje niekoľko ďalších protokolov na synchronizáciu času, no sú menej používané. Príkladom je *Precision Time Protocol* (PTP), ktorý je vhodný do lokálnych sietí kvôli vysokej presnosti. Aj napriek problémom a útokom na tento protokol nie je dobrým riešením ho zakázať, pretože aktuálny čas je v diagnostike a monitorovaní nesmierne dôležitý.



Obr. 3.8: Ilustrácia amplifikačného útoku cez nakazený počítač pomocou podvrhnutej IP adresy [22].

3.4.8 Záloha a zabezpečenie konfigurácií

Konfigurácie zariadení a ich záloha sú veľmi dôležitým faktorom, ktorým sa treba zaoberať pri správe infraštruktúry. Pokiaľ sú prítomné aktuálne konfigurácie zariadení, tak pri výpadku hardware je možné ho vymeniť za nový a aplikovať fungujúcu konfiguráciu z poškodeného zariadenia zo zálohy. Zároveň by sa konfigurácia mala dostatočne zabezpečiť proti výmazu zo zariadenia a zálohovaného úložiska a dostatočne zabezpečiť [2]. Zabezpečenie je dôležité, aby nedošlo k úniku konfigurácie k útočníkovi a nepovolaným osobám a následnému zneužitiu. Záloha konfigurácií by sa mala robiť cez zabezpečený kanál najlepšie pomocou protokolov podporujúcich šifrovanie, napríklad *Secure Copy Protocol* (SCP) alebo *Secure File Transfer Protocol* (SFTP) a nie pomocou *Trivial File Transfer Protocol* (TFTP) [8]. Vhodná je aj prítomnosť záznamu zmien v konfigurácií v čase [2].

3.4.9 Správanie pri vysokom zaťažení

V priebehu prevádzky sa môže vyskytnúť kratší alebo aj dlhý časový okamih, kedy je zariadenie vysoko zaťažené a nezvláda spracovávať požiadavky. Toto môže byť spôsobené útokom (D)DOS alebo nedostatočným dimenzovaním a zlou architektúrou siete. Aj napriek tomuto stavu by však malo byť zariadenie schopné odosielať chybové správy a notifikovať o problémoch. Zároveň by mali byť nastavené prahové hodnoty, ktoré budú indikovať stav, že môže dôjsť k nadmernému vyťaženiu procesoru, pamäti alebo linky či už pomocou Syslog správ alebo protokolu SNMP [21][8].

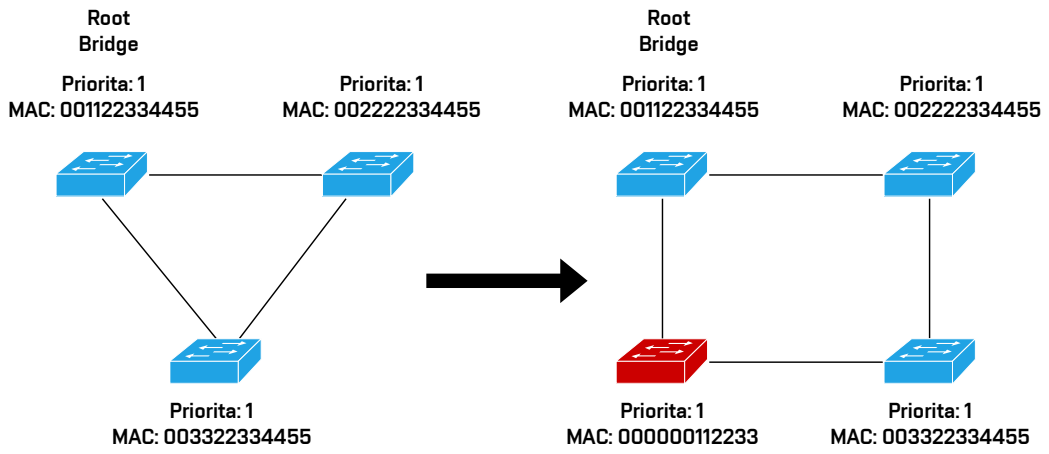
3.4.10 Monitorovanie výkonu siete

Monitorovanie siete nie je len o chybových a operačných správach zariadení, ale aj o prevádzke, ktorá v sieti prebieha. Toto monitorovanie prevádzky musí byť vykonávané často z legislatívnych dôvodov a aplikuje sa u poskytovateľov pripojenia. Monitorovanie prevádzky sa však vykonáva aj v lokálnych sieťach, napríklad zrkadlením portov [8] na analýzu útokov pre IDS alebo pre štatistické informácie a informácie o zaťažení pomocou protokolov sFlow a NetFlow.

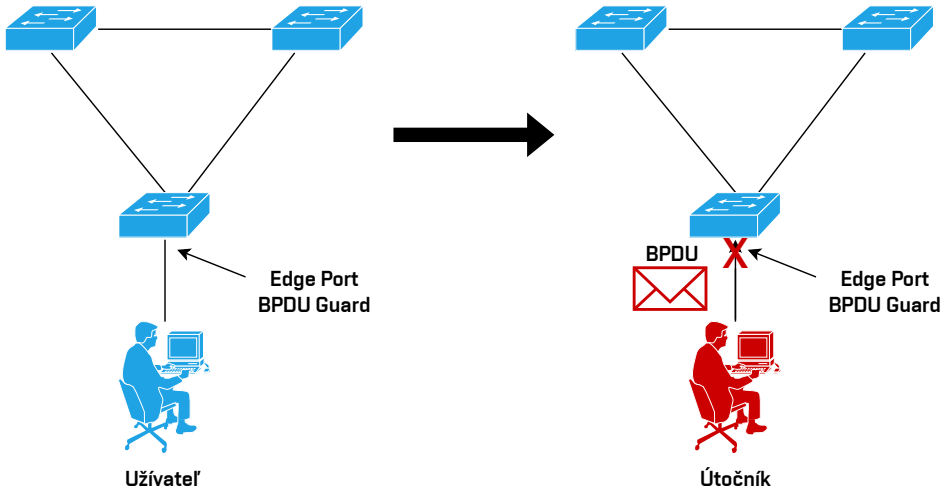
3.4.11 Problémy vrstvy L2

Prístupová vrstva hierarchického modelu alebo vrstva L2 modelu ISO/OSI je časť siete, do ktorej sa pripájajú zväčša koncové zariadenia. Vzniká tu preto mnoho problémov či už bezpečnostných alebo prevádzkových, na ktoré je nutné myslieť.

Protokolom pracujúcim na vrstve L2 je *Spanning Tree Protocol* (STP), zaisťujúci bezslučkovú topológiu aj v prípade cyklického zapojenia prepínačov z dôvodu redundancie [7]. Existuje mnoho implementácií STP protokolu, každé však zabezpečuje logické vypnutie alebo zakázanie portu aj pri existujúcom fyzickom pripojení. Pre urýchlenie výpočtu kostry a konvergencie siete sa vylučujú z výpočtu porty, na ktoré sú pripojené koncové zariadenia a teda nepredpokladá sa na týchto portoch pripojený prepínač. Tento fakt môže na druhej strane spôsobiť slučky v prípade zapojenia prepínača do takéhoto portu. Z tohto dôvodu existuje tzv. BPDU Guard [7], čo je ochrana, kedy pri prijatí rámca s BPDU označeným červeno na obrázku 3.10 na port vyradený z výpočtu kostry grafu je port zablokovaný a neumožňuje preposielať rámce. Ďalšou ochranou je Root Guard [3], ktorý zabraňuje novo pripojeným prepínačom prebrať rolu hlavného prepínača pre danú podsieť alebo VLAN. Jeho úžitok zobrazuje nasledujúci obrázok 3.9, kde po pripojení nedovoleného prepínača označeného červeno nepríde k zvoleniu nového Root Bridge kvôli ochrane Root Guard. Existuje ešte ochrana Loop Guard [3], ktorá zabezpečuje, že pri poruche a jednosmernej komunikácií medzi prepínačmi nedôjde k vytvoreniu slučiek. Táto ochrana je však výhradne u zariadení od spoločnosti Cisco.



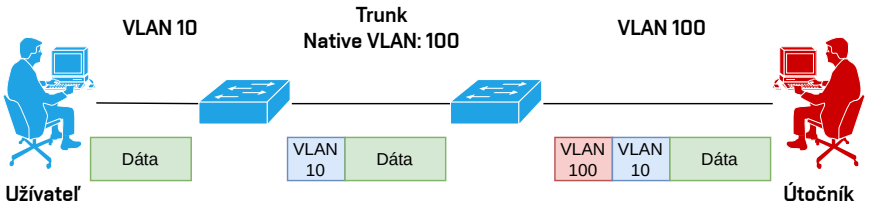
Obr. 3.9: Zabránenie prebratia role Root Bridge pomocou Root Guard, kde pri pripojení nedovoleného prepínača (označený červeno) s nižšou MAC adresou nebude prepínač zvolený za Root Bridge.



Obr. 3.10: Zabránenie vyhlásenia koncového portu ako portu k prepínáču pomocou BPDU Guard, kde po prijatí rámcu s BPDU (označeným červeno), príde k zablokovaniu portu.

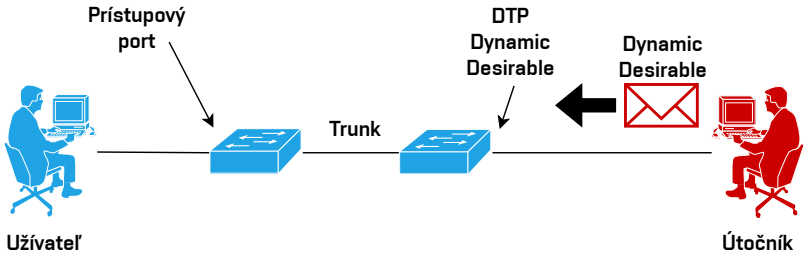
Referenčná príručka bezpečnosti [21] definuje nižšie popísané útoky na vrstve L2 a obranu na ne. V predvolenom stave sú zväčša všetky porty v jednej VLAN, ktorá je implicitne povolená na všetkých trunk portoch respektíve portoch, kadiaľ prechádza tagovaná prevádzka. Preto je dobré všetky aj nepoužívané porty odobrať z predvolenej VLAN a nepripojené porty prideliť nikam nesmerovanej VLAN. Taktiež by sa predvolená VLAN nemala používať na tagovanú prevádzku, a to aj z dôvodu, že pri zabudnutí odstránenia prístupových portov z predvolenej VLAN môže dôjsť

k útokom VLAN Hopping za pomoci Double Tagging. Na tagovaných trunk portoch by mala byť povolená prevádzka iba takých VLAN, ktoré sú potrebné a zakázaná pre VLAN, do ktorej sú umiestnené nevyužívané porty. Útoku Double Tagging sa dá zabrániť špecifikovaním VLAN na prístupových portoch a definovaním separátnej VLAN na tagovaných trunk portoch.



Obr. 3.11: Útok Double tagging, pri ktorom útočník zasiela rámec s dvoma VLAN ID, kde prvé bude odstránené prvým prepínačom, keďže trunk má tagovanú prevádzku na VLAN 100, tak dôjde k preposlaniu rámcu až k obeti [24].

Predstieranie, že koncové zariadenie je prepínač sa dá zneužitím protokolu *Dynamic Trunking Protocol* (DTP) [21]. Pri tomto útoku s názvom Switch Spoofing, prepínač aktívne alebo pasívne čaká na odpoveď, že na druhej strane prístupového portu je prepínač. Ak mu dôjde od koncového zariadenia takáto správa, tak sa prepne pôvodne prístupový port na port typu trunk. Z tohto dôvodu je dobrou zásadou tento protokol nevyužívať a porty konfigurovať ručne ako prístupové alebo trunk.



Obr. 3.12: Útok Switch Spoofing pomocou protokolu DTP, prepínač čaká na správu DTP alebo stav portu trunk, útočník zasiela správu (označenú červeno) žiadajúcu o vytvorenie trunk portu, trunk bude nakoniec vytvorený.

Istou formou zabezpečenia je aj explicitne zakázať, t.j. vypnúť nepoužívané prístupové porty, aby ich nebolo možné zneužiť, keďže často pri neaktívnych portoch absentujú rôzne bezpečnostné nastavenia, z dôvodu, že pri prvotnom nasadení zariadenia sa nevyužívali.

Proprietárny protokol *Virtual Trunking Protocol* (VTP) a štandardizovaný *Multiple VLAN Registration Protocol* (MVRP) a ich predchodcovia umožňujú distribúciu a synchronizáciu VLAN informácií na skupinu prepínačov [3]. Na jednej strane tieto protokoly uľahčujú administráciu, no pri neopatrnosti môže pri zapojení nového zariadenia do siete prísť k výmazu VLAN informácií na všetkých pôvodných prepínačoch. Preto je dobré tento protokol používať iba pri prvotnom nasadení a vytváraní siete. V prípade nutnosti používania tohto protokolu je dobré zabezpečiť správy posielané medzi prepínačmi, aby nedošlo k ich manipulácií po ceste. Taktiež je v prípade použitie týchto protokolov výhodné zapnúť funkciu pruning, ktorá umožňuje zasielať broadcast iba na tie prepínače, ktoré majú porty v danej VLAN.

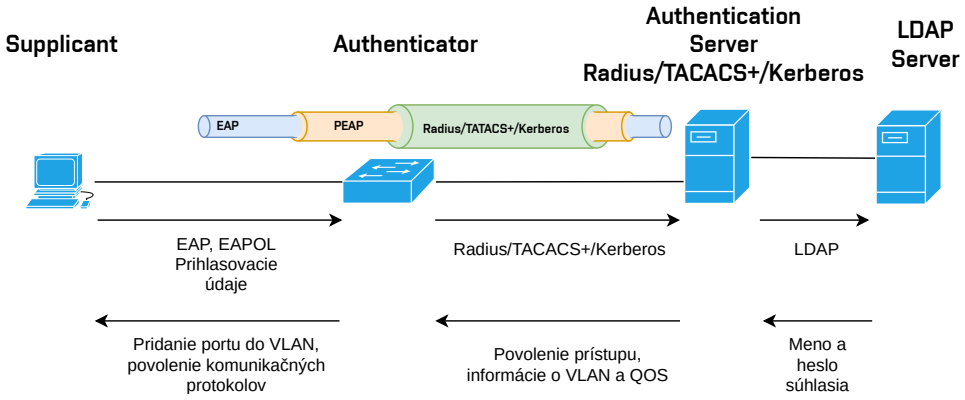
3.4.12 First Hop Security IPv4

First Hop Security je označenie pre niekoľko prístupov k zabezpečeniu koncových staníc a mitigáciu rôznych zneužití a podvrhnutí. Treba povedať, že protokoly IPv4 a IPv6 sa mierne odlišujú vzhľadom ku rozdielnemu postoju k prideľovaniu adries [25].

Prístupové porty, ktoré sa využívajú na pripojenie klientských staníc zväčša absentujú akoukoľvek identifikáciou pripojeného zariadenia. Najjednoduchším spôsobom je definovanie maximálne jednej povolenej MAC adresy na porte. Tento typ obrany naviac zamedzí útoku MAC Flooding, kde dochádza k zaplaveniu portu náhodnými MAC adresami za účelom preťažiť CAM tabuľku prepínača a donútiť ho posielať všetko ako broadcast. Pri prekročení limitu počtu adries na port by mal byť notifikovaný administrátor a port by mal pozastaviť preposielanie rámcov.

Napriek vyššie zmienenému opatreniu, nič nebráni útočníkovi zmeniť MAC adresu na útočiacom zariadení, aby mu bol povolený prístup do lokálnej siete, tento typ zneužitia sa volá MAC Spoofing. Naviac takéto nastavenie zamedzí využívanie prípojky legitímnym užívateľom. Z tohto dôvodu vznikol štandard 802.1x, ktorý definuje akým spôsobom bude užívateľ respektíve koncová stanica na porte autentizovaná [3]. Tento štandard využíva protokoly *Extensible Authentication Protocol* (EAP), *Protected Extensible Authentication Protocol* (PEAP), *Extensible Authentication Protocol over LAN* (EAPoL), RADIUS, TACACS+, Kerberos a definuje tri role pre zariadenia podieľajúce sa na autentifikácií. Prvým typom zariadenia je Supplicant, čo je koncové zariadenie, ktoré zasiela prístupové údaje na zariadenie Authenticator, ktorým je zväčša prepínač. Na porte prepínača, na ktorý je pripojený Supplicant sú bez overenia povolené len protokoly EAP, EAPOL prípadne CDP/LLDP a je umiestnený do izolovanej VLAN. Po úspešnom overení pomocou autentifikačného serveru, ktorý porovná preposlané prihlasovacie údaje od prepínača s autentifikačným serve-

rom, budú na porte povolené všetky potrebné protokoly a koncové zariadenie bude v náležitej VLAN. Nižšie uvedený zjednodušený obrázok 3.13 ilustruje komunikáciu koncovej stanice s prepínačom a autentifikačným serverom.



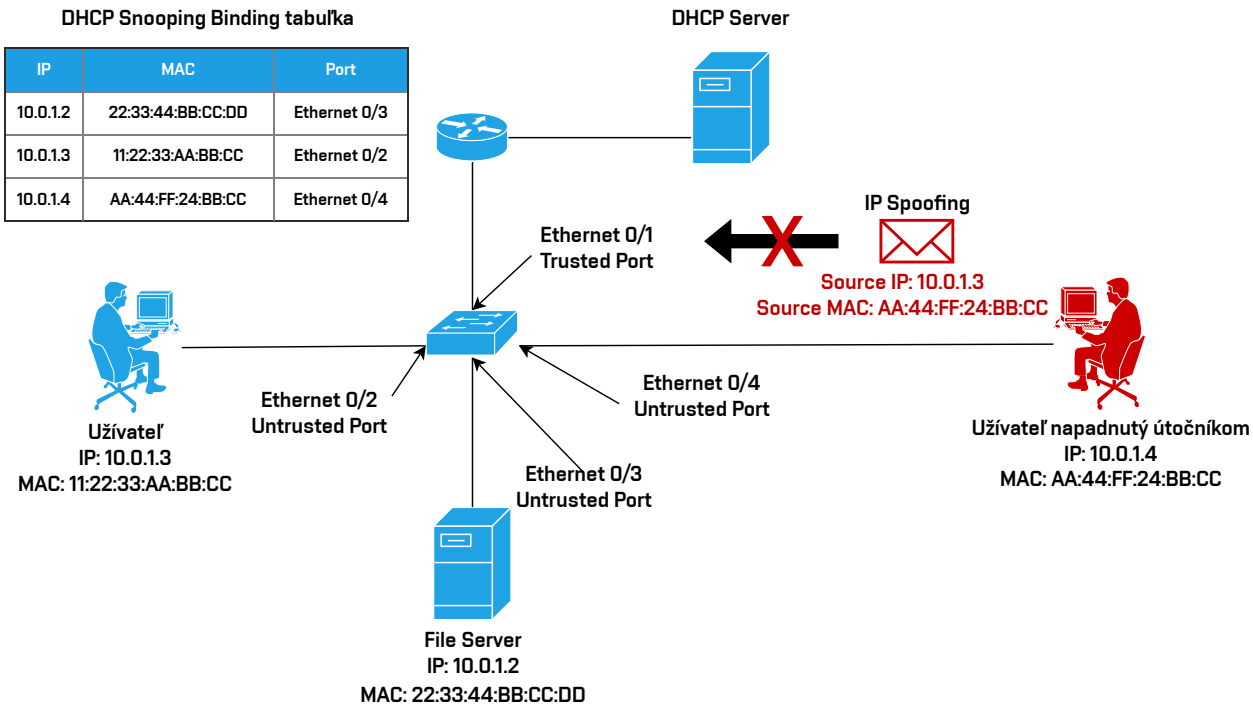
Obr. 3.13: Zjednodušená komunikácia koncovej stanice s prepínačom, ktorý preposiela prihlasovacie údaje serveru na overenie na následné povolenie komunikácie na porte [3].

Protokol IPv4 využíva centralizované prideľovanie IP adries za pomoci DHCP serveru. Výhodou je, že všetky pridelené adresy sú dostupné na jednom mieste, no zároveň toto riešenie porušuje vrstvový model sietí, keďže aplikačný protokol DHCP konfiguruje protokol nižšej vrstvy. Problém v prípade útoku DHCP Spoofing Man-in-the-middle je, že v sieti môže byť viacero DHCP serverov a ten, ktorý odpovie rýchlejšie, teda útočníkov môže prinútiť koncovú stanicu, aby si vybrala adresu ponúkanú ním. To znamená, že môže napríklad všetku sieťovú prevádzku z koncového zariadenia smerovať cez bod siete, ktorý útočníkovi vyhovuje. Obranou na tento útok je DHCP Snooping [3][8], kedy sa definujú porty, ktoré sú dôveryhodné a teda, môžeme z nich prijímať správy DHCP Offer a DHCP Acknowledge. Prepínač si následne vytvorí mapovanie IP adresy pridelenej DHCP serverom, MAC adresy koncového zariadenia, VLAN a portu, na ktorom je zariadenie pripojené.

DHCP Snooping tiež zabraňuje vyčerpaniu adries pridelených pomocou DHCP serveru, kde útočník posiela správy DHCP Request s podvrhnutými MAC adresami, aby vyčerpal rozsah, ktorý sa prideľuje klientom.

Protokol ARP býva zneužitý na útok ARP Spoofing. Pri tomto útoku útočník na dotaz užívateľa, v ktorom sa pýta, akú MAC adresu má zariadenie, ku ktorému pozná IP adresu, odpovie svojou MAC adresou alebo MAC adresou uzla v sieti, cez ktorý má prechádzať komunikácia. Týmto zabezpečí, že dáta bude možné odchytávať na stanici, ku ktorej má prístup. Obranou proti tomuto zneužitiu je Dynamic ARP Inspection [2], ktorá porovnáva údaje získané z tabuľky vybudovanej pomocou DHCP Snoopingu. Pokiaľ ARP odpoveď z portu neodpovedá naučenej informácii, tak je paket ARP odpovedi zahodený. Preto je nutnosťou pre využitie tejto obrany mať zapnutú funkciu DHCP Snooping.

Ďalším častým útokom na prístupovej vrstve je IP Spoofing, a teda používanie a zneužitie IP adresy, ktorá koncovému zariadeniu nepatrí. Útočník pomocou tohto útoku môže zahltiť stanicu tak, že ako zdrojovú IP adresu paketu uvedie IP adresu obeti a predpokladá, že odpoveď na tento paket už nebude doručená jemu, ale obeti. Mitigácia tohto útoku je možná za pomoci IP Source Guard [8], ktorá v každom odchádzajúcom pakete skontroluje, či IP adresa vysielajúcej stanice súhlasí so zdrojovou IP adresou. Nutnou prerekvizitou je vybudovanie tabuľky za pomoci DHCP Snoopingu.



Obr. 3.14: Využitie DHCP Snoopingu pre IP Source Guard, pri ktorom útočník napadne užívateľov počítač a snaží sa s podvrhnutou IP adresou zaslať paket, ten nezodpovedá mapovaniu v tabuľke na prepínači a bude zahodený.

3.4.13 First Hop Security IPv6

Protokol IPv6 používa odlišný prístup k prideľovaniu IPv6 adries, a teda nie všetky mitigácie útokov z protokolu IPv4 sú realizovateľné. V prvom rade systém prideľovanie IPv6 adries nie je povinne centralizovaný a na konfiguráciu adries sa používa protokol *Internet Control Message Protocol version 6* (ICMPv6). Je možné použiť aj DHCPv6 server, no ten nie je v štandarde definovaný ako povinný, a preto ho niektoré systémy vôbec nepodporujú. Zariadenie dostane buď od smerovača, alebo si od neho vyžiada správu Router Advertisement. V nej dostane prefix siete, prípadne ďalšie informácie a následne si spodných 64 bitov do adresy zvolí náhodne alebo si ich odvodí z MAC adresy. Keďže nie je prideľovanie centralizované a môže dôjsť k duplicite adries je treba zistiť ICMPv6 správou či zvolená adresa nie je v sieti už používaná. Práve v tomto spočíva prvý útok, kedy na správu overujúcu použitie adresy v sieti reaguje útočník, že ju má pridelenú práve on [26]. Zariadenie si vygeneruje novú adresu a pokus opakuje, útočník mu znova odpovie rovnako a teda mu odopiera prístup a znemožní mu využívať sieť. Kontrola prítomnosti adresy sa vykonáva aj v prípade využitia DHCPv6 serveru, takže ani jeho prítomnosť tento problém neodstráni.

Protokol ICMPv6 a správa Neighbor Solicitation sa v IPv6 používa namiesto protokolu ARP pre zistenie MAC adresy, teda z toho plynú podobné problémy ako pri IPv4. Čiastočná mitigácia problému je pomocou ND Inspection, no ten využíva princíp „Trust on first use“ alebo „First come first serve“, teda do tabuľky podobnej DHCP Snoopingu sa zapíše hodnota prvej prihlásenej stanice [28][29]. Tým môže útočník taktiež odstaviť užívateľa od pripojenia a zamedziť kontrolu duplicity adries, pokiaľ si svoje zariadenie prihlási ako prvé. Druhou možnosťou je DHCPv6 Snooping, no ten je komplikované realizovať v prípade staníc s rôznymi systémami kvôli nekompatibilite tohto protokolu naprieč operačnými systémami. Riešením by bol protokol *Secure Neighbor Discovery* (SEND), no ten sa kvôli komplikovanosti a problémom s licenciami nepoužíva [27].

Oznámenie smerovača môže byť podvrhnuté falošným smerovačom – Rogue RA alebo koncové zariadenie môže byť zaplavené falošnými prefixmi – RA Flood [28][29]. Z tohto dôvodu je nutné používať obranu RA Guard, ktorá definuje na ktorom rozhraní prepínača je pripojený smerovač, ktorý zasiela správy Router Advertisement, teda prefixy siete a ostatné potrebné informácie. Je to obdoba trust portu u DHCP Snoopingu. Naviac je nutné zabezpečiť, aby ohlásenie smerovača a suseda nebolo fragmentované, tak ako hovorí RFC 6980 [30].

Kontrola IPv6 Source Guard, ktorá zabraňuje podvrhnutiu IPv6 adries je funkčná iba v prípade prítomnosti DHCPv6/IPv6 Snooping a ND inspection, čo ako bolo popísané vyššie môže spôsobiť aj odoprenie služby legitímnemu užívateľovi. Preto pri použití ND inspection a DHCPv6/IPv6 Snooping je vhodné pre kritické zariadenia, napríklad servery vytvoriť na prepínačoch manuálny záznam MAC a IPv6 adresy a VLAN.

Problémom v IPv6 je aj vyčerpanie pamäte susedov, kedy sa do tejto pamäte dostávajú neexistujúce útočníkove adresy alebo je v sieti pripojených veľa staníc s priveľa IPv6 adresami na každý uzol [31][32]. Tento problém pramení vo fakte, že koncové zariadenia majú hneď niekoľko IPv6 adries. Pri útokoch z internetu na globálne adresy môžeme vyfiltrovať pomocou ACL rozsahy sietí, ktoré nie sú v lokálnej sieti pripojené, a tým zamedziť ich zapísaniu do tabuľky. V lokálnych sieťach sa môže definovať maximálna doba, po ktorú bude IPv6 adresa v zariadení zaznamenaná alebo sa staticky definuje mapovanie IPv6 a MAC adresy.

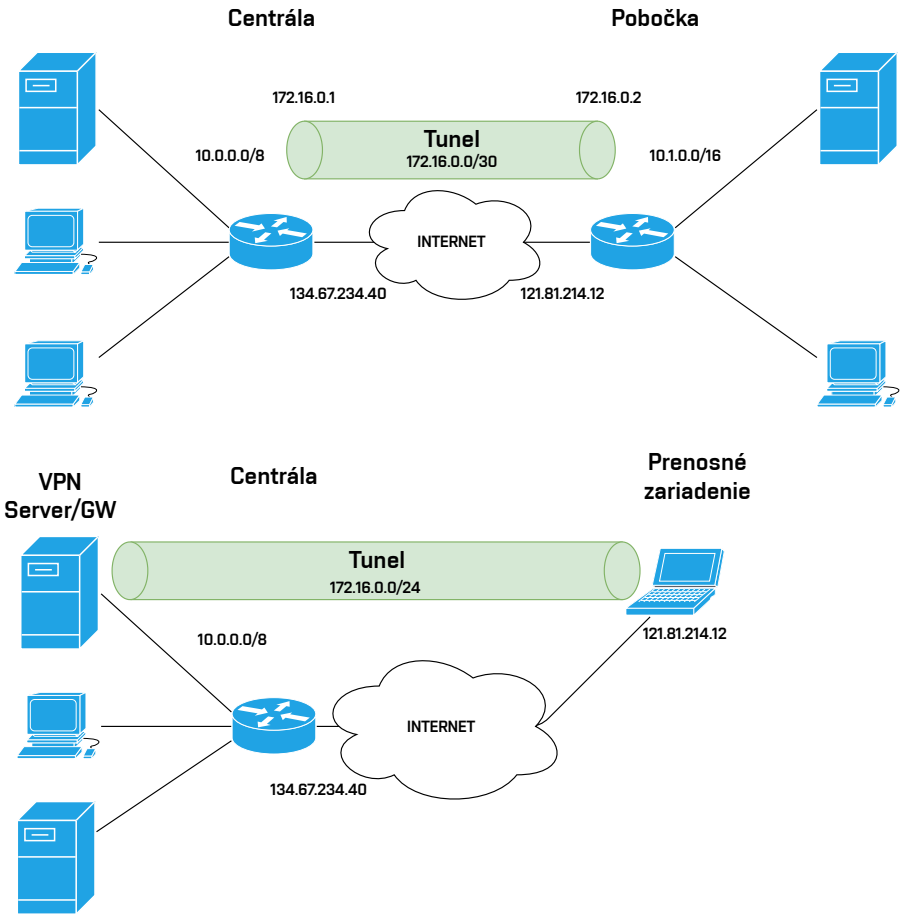
V sieťach s IPv6 konektivitou je viac než žiadúce dbať na monitorovanie, keďže všetky útoky nie je možné potlačiť.

3.4.14 First Hop Redundancy Protocols

Protokoly na redundanciu brány štandardizovaný *Virtual Router Redundancy Protocol* (VRRP) a proprietárne *Hot Standby Redundancy Protocol* (HSRP) a *Gateway Load Balancing Protocol* (GLBP) umožňujú využívať jednu virtuálnu adresu pre východziu bránu na koncových zariadeniach a tým sú pre toto koncové zariadenie transparentné [7]. Naviac proprietárny protokol GLBP dokáže na ARP dotaz vracať ktorúkoľvek MAC adresu smerovača v skupine a tým rozkladať medzi ne záťaž. Všetky protokoly umožňujú autentifikáciu správ zasielaných medzi sebou a tým istú úroveň bezpečnosti, aj keď nie úplne ideálnu.

3.4.15 Tunely a VPN

Virtual Private Network – Virtuálna privátna sieť (VPN) slúžia na vzdialené pripojenie zariadení, ktoré sú oddelené napríklad vonkajšou sieťou, internetom [7]. Pre pripojenie vzdialených zariadení sa využívajú tunely. Spravidla sa VPN rozdeľujú na dva druhy, a to site-to-site, kde je pobočka k centrále pripojená cez hraničné prvky siete pomocou permanentne vytvoreného tunelu. kde je vytvorené permanentné spojenie medzi hraničnými zariadeniami. Druhou alternatívou je remote access VPN, kedy sa vytvára tunel na vyžiadanie a všetká sieťová prevádzka môže byť smerovaná cez bod, ku ktorému sa stanica vzdialene pripája a zároveň je zariadeniu prístupná vnútorná sieť. Tieto tunely môžu byť šifrované, čo zabezpečuje dôvernosť a preto by mali byť preferovanou alternatívou. Dnes ešte stále používané protokoly *Point-to-Point Tunneling Protocol* (PPTP), *Layer 2 Tunneling Protocol* (L2TP) nie sú v dnešnej dobe považované za bezpečné. Preto sa dnes využívajú tunely pomocou protokolu *IP Security* (IPSec), prípadne pre remote access VPN je to protokol OpenVPN pracujúci na aplikačnej vrstve.



Obr. 3.15: Porovnanie site-to-site a remote access VPN

3.4.16 Mapovanie siete a objavovanie zariadení

Protokoly objavujúce zariadenia ako LLDP a *Cisco Discovery Protocol* (CDP) umožňujú získanie mnohých informácií o susedných pripojených zariadeniach, ako napríklad IP adresy, informácie o VLAN, operačnom systéme a mnohé ďalšie. Na tieto protokoly existuje veľké množstvo útokov s veľmi závažnými následkami. Častokrát sa tieto protokoly používajú pri IP telefónii a preto ich nie je možné plošne vypnúť, ideálne by sa mali zakázať aspoň na rozhraniach, kde nepotrebujú operovať.

Získavanie smerovacích informácií a masku podsiete je možné aj pomocou správ *Internet Control Message Protocol* (ICMP) typu redirects a mask reply. Problémom je aj directed broadcast, ktorý umožňuje získať ICMP odpoveď na správu ICMP Echo zaslanú na broadcast adresu smerovača. Zariadenia od spoločnosti Cisco majú túto funkciu už dlhšiu dobu z bezpečnostných dôvodov zakázanú.

Mapovanie siete je možné aj pomocou *Multicast Listener Discovery* (MLD) a *Internet Group Management Protocol* (IGMP) Query správ, prípadne správami ICMP Echo na adresu ff02::1 a 224.0.0.1 [34][33]. Na zabránenie tohto útoku je možné použiť pravidlá v ACL.

Bezpečnostným problémom, ale aj systémom porušujúci fakt, že smerovač oddeľuje siete a broadcast doménu je proxy ARP. Tento systém umožňuje preposielanie ARP správ smerovačom do ďalších sietí. Využíva sa napríklad aj pri VPN, kedy chceme spojiť dve siete na vrstve sieťového rozhrania.

3.4.17 Nepoužívané a nebezpečné služby

Sieťové zariadenia sa často predávajú s rôznymi spustenými službami a tieto predvolené nastavenia, ktoré nie sú potrebné, môžu byť terčom útokov, a preto by mali byť vypnuté [10]. Keďže administrátor tieto funkcie nepoužíva, tak im ani nevenuje pozornosť pri zabezpečovaní. Typickými príkladmi sú administrácia pomocou protokolu *Hypertext Transfer Protocol* (HTTP) prípadne spustený HTTP server a podobne.

3.4.18 Ostatné bezpečnostné a prevádzkové postupy

Pre korektné fungovanie viacerých protokol je vhodné využívať ako zdroj Loopback rozhranie. Preto je dobrým zvykom definovať jedno Loopback rozhranie na zariadení, ktoré je dostupné hneď po štarte, nie ako fyzické rozhrania a môže byť užitočné ako identifikátor zariadenia pre viaceré protokoly. Toto rozhranie respektíve IP adresa sa používa ako zdrojová pri protokoloch *Network Time Protocol* (NTP), RADIUS, Tacacs+, SNMP, Syslog, SSH a tiež k identifikácií staníc dynamických smerovacích protokolov [5][8][10].

4 Návrh

4.1 Požiadavky na aplikáciu a existujúce riešenia

V súčasnosti existuje zopár programov funkcionalitou podobných vytváranému. Jedným z príkladov je open-source riešenie `Cisco Config Analysis Tool` [51], ktorý čerpá odporúčania z jednej literatúry [8], pomocou ktorej boli vytvorené aj odporúčania v tejto práci. V tomto riešení však chýba veľa dôležitých prevádzkových a bezpečnostných odporúčaní z dôvodu, že námetom na kontrolný zoznam pri zostavovaní aplikácie bola iba jedna literatúra. Taktiež podporuje iba jedného výrobcu sieťových zariadení a chýba mu modularita, nerozlišuje odporúčania a kontrolu ich prítomnosti na základe umiestnenia sieťového zariadenia v hierarchickom modeli. Existujúci nástroj `Cisco Config Analysis Tool` obsahuje kontrolu iba na niektoré bezpečnostné nastavenie pre protokol IPv6. Rozšíriteľnosť tohto riešenia je takmer nulová vzhľadom na absenciu modularity.

Druhým nástrojom s podobnými vlastnosťami a nedostatkami je aj `Router Auditing Tool` [52], ktorý má naviac aj *Graphical User Interface – grafické užívateľské rozhranie* (GUI). Existuje niekoľko rozšírení aj pre nástroj `Nessus`, ktoré overujú dodržiavanie odporúčaní a podľa zistení čerpajú z CIS Benchmarku [10]. Taktiež však nepodporujú zjednanie nápravy a ignorujú umiestnenie zariadenia v topológii.

Kľúčovou vlastnosťou vytváraného programu je modularita, vďaka ktorej bude možné pridávať a definovať nové moduly na odhaľovanie a opravu nedostatkov alebo meniť existujúce pri zmene syntaxe a sémantiky príkazov. Modularita taktiež umožňuje vytvorenie a podporu ďalších výrobcov a operačných systémov sieťových zariadení. Existujúce riešenia sú zväčša zamerané iba na jedného výrobcu a operačný systém, pričom program je jeden zdrojový súbor, ktorý bez dobrej znalosti kódu je problematické upraviť a rozšíriť. Preto jednotlivé overovania odporúčaní a ich následná oprava bude každé v separátnom module, ktorý budú musieť dodržať určité vstupy a výstupy, teda akési *Application Programming Interface* (API). Existujúce riešenia nedisponujú žiadnym generovaním opravnej konfigurácie na základe nálezu nedostatku, preto vzniknutá aplikácia bude podporovať aj vygenerovanie nápravy.

Výhodou výsledného programu je aj, že kontrolný zoznam vznikol z viacerých knižných odporúčaní a benchmarkov organizácií zaoberajúcimi sa danou problematikou. Program bude umožňovať spúšťanie modulov zodpovedných za nájdenie a odstránenie nedostatkov na základe definovaného umiestnenia zariadenia v hierarchickom modely siete. Tým sa zamedzí generovaniu falošne pozitívnych správ, ktoré by vznikli v dôsledku overovania nerelevantných požiadavkov na zariadenie v danej

vrstve modelu. V neposlednom rade bude riešenie zdarma s možnosťou nahliadnuť a modifikovať, respektíve rozšíriť kód. Program bude umožňovať kontrolu odporúčaní aj pre protokol IPv6, ktorým sa príliš nezaoberá väčšina odporúčaní a benchmarkov.

Kľúčové vlastnosti:

- Modularita – každé overenie s nápravou bude v zvlášť súbore prihliadajúc na výrobcu a operačný systém, pre ktoré je určené.
- Prispôsobenie na ďalších výrobcov – definovanie API pre moduly na budúcu podporu pre zariadenia od viacerých výrobcov a ich operačných systémov.
- Zjednanie nápravy – pri nájdení nedostatku automatické vygenerovanie opravného nastavenia, pokiaľ je to možné. Náprava je generovaná, ak sú definované nutné premenné pre príkaz zjednávajúci nápravu alebo charakter nastavenia umožňuje automatické generovanie príkazu.
- Podpora IPv6 – detekcia zlého alebo chýbajúceho nastavenia a následná náprava nielen pre nastavenia využívajúce najviac rozšírený protokol IPv4, ale aj pre protokol IPv6.
- Rozdelene odporúčaní podľa vrstvy zariadenia, na ktorom majú byť nastavené
- Hierarchický model – skenovanie nedostatkov typických pre jednotlivé vrstvy, v ktorých sa zariadenia nachádzajú a tým zníženie falošne pozitívnych varovaní.
- Definovanie závažnosti – každý nájdený nedostatok je hodnotený na 4 stupňovej škále.
- Personalizácia – definovanie modulov, ktoré sa spustia pre jednotlivé vrstvy, zmena závažnosti nájdených nedostatkov v konfiguračných súboroch.
- Zoznam útokov a problémov aktuálne bežiacej verzie operačného systému.
- Automatické vygenerovanie správy s nedostatkami a ohodnotenia bezpečnosti pomocou skóre pre každé zariadenie.

4.2 Rozdelenie príkazov

Na zariadeniach od firmy Cisco s operačným systémom IOS bol vykonaný rozbor možných príkazov a ich foriem zápisu a početnosti výskytu v konfigurácií. Tento rozbor bol spravený z dôvodu, že niektoré príkazy sa môžu opakovať a zároveň jeden druh príkazu môže byť konfigurovaný v rôznych kontextoch a teda neprítomnosť v jednom kontexte automaticky neznamená nedostatok v konfigurácií. Umožňuje to generalizovať príkazy do skupín a nebude nutné pre každý príkaz písať zdrojový kód, ktorý by riešil nález a nápravu iba pre konkrétny jeden prípad. Na základe rozboru boli rozdelené príkazy na konfiguráciu sieťových zariadení do nasledujúcich kategórií:

1. Maximálne s jedným výskytom v konfigurácii – príkladom môže byť verzia protokolu SSH.

Výpis 4.1: Konfigurácia verzie protokolu SSH

```
Router(config)#ip ssh version 2
```

2. Viacnásobný výskyt – typickým príkladom je definícia lokálnych účtov, ktoré môžu byť nastavené aj s nezahašovaným heslom. Takéto nastavenie sa môže vyskytovať aj viackrát, preto je potreba skontrolovať každý výskyt. V prípade lokálneho účtu sa musia eliminovať všetky účty s nezahašovaným heslom, nie iba prvý výskyt. Pokiaľ by sa hľadalo nastavenie, ktoré je potrebné mať v konfigurácií, tak je postačujúce pri jeho absencii vygenerovať iba jednu nápravu, takýmto príkladom je konfigurácia AAA serveru. Zasa platí, že dané nastavenie môže byť prítomné viackrát, teda môžeme mať definovaný aj záložný server.

Výpis 4.2: Konfigurácia účtu s nezahašovaným heslom

```
Router(config)#no username test_user password AAA
Router(config)#no username test_user2 password BBB
```

Výpis 4.3: Konfigurácia AAA serveru

```
Router(config)#radius server RAD_SERV
Router(config-radius-server)#address ipv4 10.0.0.8
Router(config-radius-server)#key PASSWD
Router(config)#tacacs server RAD_SERV
Router(config-tacacs-server)#address ipv4 10.0.0.9
Router(config-tacacs-server)#key PASSWD
```

3. Viacnásobný výskyt viazaný na rozhranie – typickým príkladom je zabezpečenie portu s definovaním maximálneho počtu povolených *Media Access Control* (MAC) adries.

Výpis 4.4: Konfigurácia maximálneho počtu povolených MAC adries na porte

```
Router(config)#interface FastEthernet0/1                          1
Router(config-if)#switchport port-security mac address max 1      2
Router(config)#interface FastEthernet0/4                          3
Router(config-if)#switchport port-security mac address max 2      4
Router(config)#interface FastEthernet1/1                          5
Router(config-if)#switchport port-security mac address max 1      6
```

4. Viacnásobný výskyt v rôznych kontextoch – tieto príkazy konfigurujú rôzne služby, napríklad autentifikáciu správ OSPF alebo prístup k manažmentu zariadenia.

Výpis 4.5: Konfigurácia autentizácie OSPF na porte alebo v procese

```
Router(config)#interface FastEthernet0/1                          1
Router(config-if)#ip ospf message-digest-key 1 md5 heslo          2
Router(config-if)#ip ospf authentication message-digest           3
Router(config)#router ospf 1                                      4
Router(config-router)#area 0 authentication message-digest        5
Router(config-router)#area 0 authentication key-chain 1           6
```

Výpis 4.6: Konfigurácia SSH prístupu na zariadanie

```
  Router(config)#line vty 0 4                                     1
  Router(config-line)#transport input ssh                         2
  Router(config)#line vty 5 6                                     3
  Router(config-line)#transport input ssh                         4
```

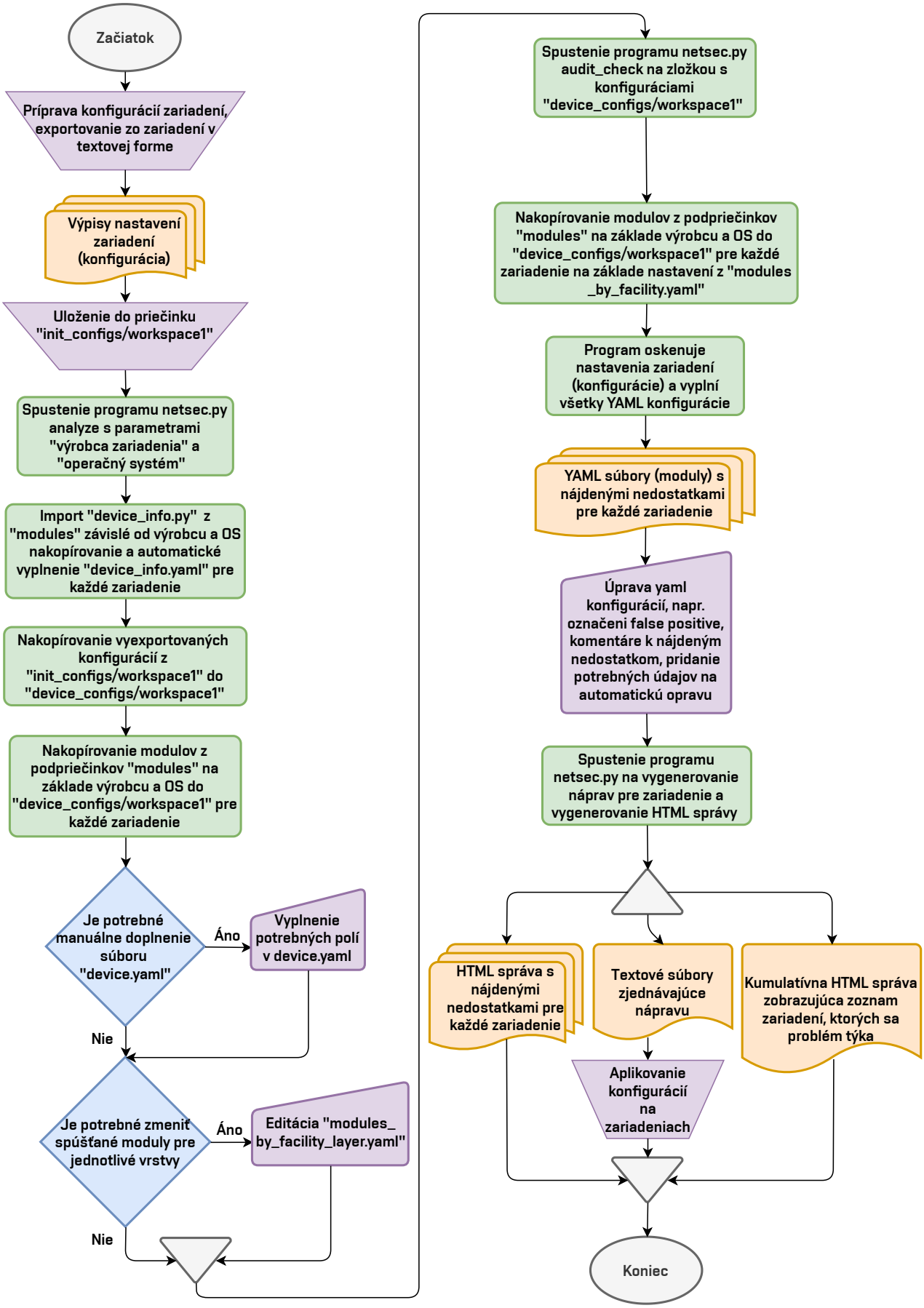
4.3 Rozdelenie sieťových prvkov

Sieť je dnes navrhovaná zväčša podľa hierarchického modelu opísaného v kapitole 3.2. Preto sa aj problémy a útoky v návrhu zatrieďujú podľa vrstvy, ktorú ovplyvňujú. V praxi sa však v menších sieťach funkcie jednotlivých vrstiev zlučujú, a preto boli okrem štandardných vrstiev nad rámec hierarchického modelu definované nasledujúce:

- Core – core vrstva, prípadne s funkciou hraničného prvku.
- Distribution – distribučná vrstva.
- Access – prístupová vrstva.
- Collapsed All – všetky vyššie zmienené vrstvy zlúčené do jednej.
- Collapsed Distribution Access – zlúčená distribučná a prístupová vrstva.
- Collapsed Core Distribution – zlúčená core a distribučná vrstva.

4.4 Princíp fungovania

Vstup programu počíta s vopred vyexportovanými konfiguráciami v textovej podobe, tie sú často dostupné z pravidelných záloh. Môže sa stať, že nie všetky nastavenia potrebné na dôsledné oskenovanie sú k dispozícií a preto je možné vyexportovať potrebné výstupy zo zariadenia pomocou skriptu, ktorý by bol spustiteľný na zariadení. Program umožní automaticky vyplniť polia v konfiguračnom súbore pre zariadenie `device.yaml`, no v prípade nepostačujúcich informácií zo stiahnutých konfigurácií je potreba manuálne vyplniť niektoré parametre, napríklad vrstvu, na ktorej zariadenie pracuje. Po úspešnom oskenovaní bude možné označiť nálezy ako falošne pozitívne a okomentovať tento dôvod, následne po takomto označení nebude generovaná náprava na nález na danom zariadení. Výstupom bude správa v HTML formáte s nedostatkami spustiteľná v akomkoľvek prehliadači, s informáciou o zjednanej náprave alebo označení falošne pozitívneho nálezu. Aplikovanie nápravy bude zjednané manuálne nakopírovaním tejto vygenerovanej nápravy na zariadenie prípadne pomocou programov na hromadné nasadenie ako napríklad Ansible, ktorý mnohé spoločnosti využívajú k hromadnému nasadzovaniu.



Obr. 4.1: Zjednodušený vývojový diagram opisujúci prácu s programom a tok dát v programe

Hierarchická štruktúra

Nasledujúca štruktúra súborov s komentármi vyobrazuje hierarchiu uloženia potrebných súborov na vykonávanie skenovania nedostatkov a zjednanie náprav. Nové konfigurácie stiahnuté zo zariadenia sa schválne nedávajú do zložky `device_configs`, keďže táto zložka môže obsahovať už nejakú nápravu a správu k predchádzajúcej konfigurácie, a preto by mohol vzniknúť konflikt a nebolo by jasné ku ktorej verzii konfigurácie výstupy programu patria. Program bude uchovávať taktiež históriu jednotlivých skenovaní a náprav, pričom pri iniciovaní nového skenovania a prítomnosti novej verzie konfigurácie zariadenia, príde k premiestneniu aktuálnych výstupov do podzložky `old_configs` s časovým razítkom v názve adresára. Moduly, respektíve šablóny YAML súborov definujúce nájdenie problému sú uložené hierarchicky na základe výrobcu sieťového zariadenia a následne v podzložke operačného systému. Príkladom cesty takto uložených modulov je sekvencia `cisco/ios/yaml_module_config`. Popis jednotlivých súborov a adresárov je v stromovej štruktúre nižšie.

- `/` ..... koreňový adresár programu
  - `netsec.py` ..... hlavný program
  - `modules` ..... adresár so šablónami konfiguračných súborov a modulmi
    - `cisco` ..... výrobca sieťového zariadenia
      - `ios` ..... operačný systém bežiaci na zariadení
        - `device_info.yaml` . šalóna na uloženie základných informácií o zaridení
        - `yaml_module_configs`
          - `01_01_aaa_new_model.yaml` ..... šablóna konfiguračného súboru k modulu
          - `11_02_cdp_enable.yaml` .. šablóna konfiguračného súboru k modulu
        - `python_modules`
          - `device_info.py` ..... Python modul/trieda na uloženie informácií o zariadení a prácu s device_info.yaml
        - `modules_by_facility_layer.yaml` . šablóna s definíciou, ktoré moduly budú spustené pre zariadenia na konkrétnej vrstve modelu
    - `hp`
    - `juniper`
      - `junos`
    - `device_info_abstract.py` ..... abstraktná trieda z ktorej dedia všetky nové device_info.py
    - `own_variables.yaml` .... šablóna pre premenné nutné pri generovaní nápravy
    - `yaml_module.py` ..... Python modul/trieda na prácu s YAML konfiguračnými modulmi
  - `device_configs` ... adresár s výstupmi, konfiguráciami a súbormi na ich nápravu
    - `workspace1` ..... adresár jednej topológie/siete
      - `current_fixes` ..... adresár s nápravami

- hostname1_fix.txt
- hostname2_fix.txt
- reports ..... adresár so správami
  - hostname1_report.html .. detailnejšia správa pre konkrétne zariadenie
  - hostname2_report.html
  - summary_report.html ..... správa s mapovaním nedostatok-zariadenie
- hostname1 ..... zložka pre jedno konkrétne zariadenie
  - device_info.yaml ..... základané informácie o konkrétnom zariadení
  - 01_01_aaa_new_model.yaml konfiguračný súbor k modulu s nájdenými nedostatkami
  - 11_02_cdp_enable.yaml
  - hostname1_config.txt ..... aktuálne skenovaná konfigurácia
- hostname2
- own_variables.yaml súbor s premennými nutnými pri generovaní nápravy
- modules_by_facility_layer.yaml .súbor s definíciou, ktoré moduly budú spustené pre zariadenia na konkrétnej vrstve modelu

- workspace2

- init_configs ..... adresár s novými skopírovanými konfiguráciami zariadení
  - workspace1
    - hostname1_config.txt
    - hostname2_config.txt
  - workspace2
- old_configs ..... adresár s predchádzajúcimi nálezmi a nápravami z rovnakého workspace
  - workspace1
- report ..... adresár s CSS a HTMLpre vytvorenie správy
  - report.css
  - report_begin.html
  - report_end.html

4.5 Zoznam odporúčaní

V súčasnej dobe existuje mnoho odporúčaní, štandardov a benchmarkov, ktoré sa zaoberajú bezpečnosťou a správnou konfiguráciou sieťových zariadení. V mnohých prípadoch sú buď príliš všeobecné, a teda sieťoví inžinieri majú problém zistiť, čo daným odporúčaním autor myslel a ako ho implementovať, alebo sú určené iba pre zariadenia od jedného výrobcu. Problémom je taktiež, že väčšina odporúčaní, štandardov a benchmarkov sa nie úplne prekrývajú, a teda je potrebné pri nastavovaní a audite zariadení čerpať s mnohých naraz. Nižšie uvedené výsledné tabuľky obsahujú odporúčania z odbornej literatúry, štandardov a benchmarkov verejne dostupných a používaných v produkčnom nasadení. Výhodou je aj fakt, že obsahujú odporúčania vychádzajúce z problémov IPv6, ktoré nie sú často v štandardoch a benchmarkoch dostupné.

Zariadenia Cisco boli pre túto prácu vybrané z dôvodu, že spoločnosť Cisco je lídrom v sieťových zariadeniach, ktorý udáva trend, ich zariadenia sú celosvetovo v korporáciách veľmi rozšírené a mnoho literatúry a benchmarkov sa odvoláva na nastavenia týchto prístrojov s udávanými príkladmi konfigurácie. Taktiež sú tieto zariadenia dobrým referenčným príkladom pre hľadanie alternatívy v zariadeniach od iných výrobcov.

Tabuľky sú rozdelené do skupín podľa protokolu prípadne okruhu problému, ktorému sa venujú. V tabuľkách je možné vidieť, že odporúčania sú zatriedené podľa viacerých kritérií.

Stĺpec závažnosť (severity) vznikol na základe predpokladaných závažností. Tento atribút bude možné zmeniť v konfiguračnom súbore každého modulu v závislosti na riziku, ktoré sa pre danú topológiu a firmu vyhodnotí za pomoci manažmentu rizík opísaného v kapitole 2. Tento atribút sa nenachádza v žiadnom štandarde ani benchmarku, z ktorého vytvorený zoznam odporúčaní čerpal, no je veľmi dôležitý z hľadiska, že nie všetky nedostatky sú rovnako závažné a nemajú rovnaký dopad. Hodnoty, ktoré nadobúda sú prebrané zo štandardu CVSS, pričom posledný interval `none` reprezentujúci nulové riziko respektíve závažnosť je zamenený za kľúčové slovo `notice`. K tejto zmene prišlo z dôvodu, že problémy s nulovým rizikom nie sú súčasťou návrhu a nemá zmysel ich riešiť. V prípade, že bude nález falošne pozitívny alebo riziko bude akceptované, tak sa táto skutočnosť uloží do konfiguračného súboru. Závažnosť `notice` bude použitá v prípade prítomnosti monitorovania portu pomocou zrkadlenia portu alebo NetFlow/sFlow. Jedná sa totiž o technológie potrebné na monitorovanie prevádzky z legislatívnych alebo bezpečnostných dôvodov. Riziko existuje iba pri nesprávnom nastavení zdrojov monitorovania a cieľu pre zber dát, a preto je dobré vedieť pri audite o prítomnosti tohto nastavenia.

Posledným rozdelením je vrstva, na ktorej zariadenie pracuje (Facility layer),

nakoľko rozdelenie podľa zariadení nie je dostatočné, pretože napríklad L3 prepínač môže byť použitý na ktorejkoľvek vrstve hierarchického modelu a každá vrstva má určité špecifiká, ktoré neobsahuje iná vrstva. Špeciálny YAML konfiguračný súbor bude obsahovať informáciu, do ktorej vrstvy (Facility layer) daný modul patrí a na základe toho bude môcť program rozhodnúť, ktoré moduly zodpovedné za nájdenie problému a jeho vyriešenie budú nad vyexportovaným konfiguračným súborom zariadenia spustené. Taktiež bude možné meniť, dopĺňať a zakázať spúšťanie modulov pre jednotlivé zariadenia, pokiaľ by v danej topológii nevyhovovalo rozdelenie z tabuliek uvedených nižšie.

Vrstva, na ktorej zariadenie operuje, ako aj definované zariadenie, ktorého sa odporúčanie a opatrenie týka, nie sú súčasťou žiadneho kontrolného zoznamu, benchmarku ani štandardu, z ktorého bolo čerpané. Sieťový administrátor by preto musel sám vyvodiť záver, ktoré odporúčania a postupy bude aplikovať na jednotlivé zariadenia a vrstvy hierarchického modelu. Preto vytvorená tabuľka odporúčaní už obsahuje aj zoznam zariadení, ktorých sa opatrenie týka.

Vytvorená sústava tabuliek 4.1 – 4.13 je značne dlhá. Vzhľadom k jej dôležitosti a značnému úsiliu pri jej vytváraní bolo rozhodnuté ju ponechať v hlavnom texte práce, nakoľko jej vytvorenie je neoddeliteľnou súčasťou práce. Odvíja sa od nej totiž aj implementácia ako aj prípadné budúce rozšírenie na zariadenia ďalších výrobcov a pridaní podpory pre ne.

V nižšie uvedených tabuľkách sa nachádzajú tieto skratky:

- Plane: M – Management, C – Control, D – Data
- Severity: C – Critical, H – High, M – Medium, L – Low, N – Notify
- Facility layer: A – Všetky zariadenia, CE – Core/Edge, DI – Distribution, AC – Access, CAL – Collapsed All, CDA – Collapsed Distribution Access, CCD – Collapsed Core Distribution

Tab. 4.1: Odporúčania k prístupu na manažment zariadení

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Nepovolený prístup k manažovaniu zariadenia	Vytvoriť a aplikovať ACL pre Telnet, SSH a pod. a zaznamenať v logu prístupy	M	C	A	[45] [10]
2	Neautorizovaný prístup cez nepoužívané a nezabezpečené protokoly na manažment zariadení	Vypnúť nepoužívané protokoly na prístup k manažovaniu zariadení (telnet a pod.)	M	H	A	[10] [8]
3	Nepovolený prístup k manažmentu konfigurácie zariadenia	Vypnutie odchádzajúcich spojení pre protokoly na manažment zariadení pokiaľ sa nepoužívajú (telnet a pod.)	M	H	A	[8]
4	Prístup bez požadovaných prístupových údajov	Nakonfigurovanie protokolov na manažment zariadení, aby požadovali prístupové údaje (telnet a pod.)	M	C	A	[10]
5	Nekonzistencia konfiguračných súborov pri zmenách konfigurácie viac ako jedným administrátorom	Povoliť súčasne iba jednému administrátorovi vykonávanie zmien v konfigurácii	M	H	A	[8]
6	Nepoužívanie zabezpečeného protokolu na manažment zariadení môže viesť k odposluchu	Zapnutie SSH	M	C	A	[10] [20]
7	Nebezpečná verzia 1 protokolu SSH	SSH verzia 2	M	C	A	[2]
8	Útok na krátky RSA kľúč	Dĺžka RSA kľúča minimálne 2048 bitov	M	C	A	[10] [11]
9	Dlhé neaktívne sedenie môže byť zneužité alebo aj fyzický prístup útočníka k aktívnemu sedeniu môže viesť k zmene konfigurácie	SSH čas vypršania sedenia	M	M	A	[10] [20]
10	Hádanie hesla k RSA kľúču	SSH maximálny počet neúspešných pokusov	M	H	A	[46] [39]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
11	Útok hrubou silou na zistenie prihlasovacích údajov	Špecifikovať čas po ktorý nie je možné po N pokusoch sa prihlásiť	M	H	A	[46] [39]
12	Prihlásenie na zariadenie nie je možné kvôli zablokovaniu pre príliš veľa neúspešných pokusov	Povolenie prístupu administrátorovi na základe IP adresy, keď je protokol na manažovanie zariadení nedostupný kvôli DOS útoku	M	M	A	[46] [39]
13	Dlhé neaktívne sedenie môže byť zneužité alebo aj fyzický prístup útočníka k aktívnemu sedeniu môže viesť k zmene konfigurácie	Čas vypršania sedenia pre protokol na manažovanie zariadení	M	M	A	[10] [20] [21]
14	Možné prihlásenie do zariadenia cez telnet keď je prítomné SSH	Zakázať telnet ak je SSH aktívne	M	C	A	[10] [20]
15	Útočník nie je informovaný o právnych následkoch	Právne upozornenie pri prístupe k zariadeniu	M	L	A	[2] [10] [20]
16	Nepovolená zmena konfigurácie zariadenia	Vytvorenie hesla na editovanie konfigurácie zariadenia	M	C	A	[10] [20]
17	Nepovolený prístup k manažmentu konfigurácie zariadenia	Lokálne zabezpečené účty	M	C	A	[8] [10]
18	Centrálna správa prihlásení a dohľadateľnosť zmien v konfigurácií	Definovanie a povolenie AAA serveru na prihlásenie a definovanie záložného prihlásenia	M	H	A	[45] [10] [2] [20]
19	Centrálna správa prihlásení a dohľadateľnosť zmien v konfigurácií	Definovanie a povolenie AAA serveru na editáciu konfigurácií a definovanie záložného prihlásenia	M	M	A	[10] [20]
20	Hádanie prístupových údajov	Definovanie maximálneho počtu neúspešných pokusov o prihlásenie a následné zablokovanie účtu	M	H	A	[10] [20]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
21	Prihlásenie bez prihlasovacích údajov	Zakázať záložné prihlásenie bez poskytnutia autentifikačných prostriedkov	M	C	A	[8]
22	AAA používa primárne lokálne účty namiesto centralizovaných na serveri	AAA nesmie používať ako prvú možnosť prihlásenia lokálny účet	M	H	A	[10] [20]
23	Používateľ prihlásený do zariadenia môže spúšťať akékoľvek príkazy	Nastavenie AAA autorizácie pre spúšťanie príkazov. V prípade výpadku AAA serveru, bude užívateľ odhlásený a následne prihlásený podľa záložného prihlásenia, aby mu nebolo pridelené vysoké oprávnenie umožňujúce vykonávať príkazy, na ktoré nemá právo	M	H	A	[20] [8]
24	Administrátor vloží zlý príkaz a po čase je ho nemožné dohľadať a zjednať nápravu	Nastavenie AAA účtovania respektíve logovania pripojení a vykonaných príkazov	M	H	A	[10]
25	AAA zdrojové rozhranie nie je rovnaké pri každom reštarte	Definovanie loopback zdrojového rozhrania pre AAA	M	M	A	[10]
26	SSH zdrojové rozhranie nie je rovnaké pri každom reštarte	Definovanie loopback zdrojového rozhrania pre SSH	M	M	A	[10]
27	DOS útok na štandardný SSH port 22	Špecifikovanie iného portu pre SSH ako štandardného alebo aplikovanie Port Knocking [47]	M	H	A	[47]

Tab. 4.2: Odporúčania pre smerovanie

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Vloženie a manipulácia so smerovacími informáciami	Autentifikácia smerovacích protokolov (nie heslá v otvorenej podobe)	C	H	CE, DI, CCD, CDA, CAL	[2] [8] [10]
2	OSPF virtuálne linky degradujú výkon	Vypnutie virtuálnych liniek pre OSPF	C	H	CE, DI, CCD, CDA, CAL	[50]
3	Koncové zariadenie, užívateľ a útočník môžu vidieť smerovacie správy a topológiu siete alebo pripojenie škodlivého zariadenia, ktoré vysielať a prijímať smerovacie správy	Špecifikovanie rozhraní, ktoré nebudú prijímať smerovacie informácie	C	H	CE, DI, CCD, CDA, CAL	[18]
4	Nemožnosť sprevádzkovať procesy smerovacích protokolov v určitých prípadoch pri použití IPv6	Špecifikovanie identifikátorov smerovacích protokolov pre každý router (router ID)	C	M	CE, DI, CCD, CDA, CAL	[49] [7]
5	Vysledovateľnosť nefunkčnosti smerovacieho protokolu a nesprávneho nastavenia	Zaznamenanie zmeny v logu pri zmenách v smerovaní	C	M	CE, DI, CCD, CDA, CAL	[21]
6	Škodlivé vloženie smerovacích informácií do BGP, vzdialený útok	TTL security	C	H	CE, DI, CCD, CDA, CAL	[18] [8]
7	Nesprávne smerovanie kvôli sumarizácií	Vypnutie automatickej sumarizácie smerovacích protokolov	C	H	CE, DI, CCD, CDA, CAL	[7]
8	DOS útok na stanicu, cez ktorú bola špecifikovaná cesta a teda nemožnosť komunikácie s koncovým bodom. Alebo zosnovanie MITM útoku	Vypnutie IP source routing	C	C	CE, DI, CCD, CDA, CAL	[10]
9	DOS útok pomocou podvrhnutej IP adresy alebo vzdialený útok na smerovací protokol	Zapnutie reverse path forwarding strict/loose mode	C	H	CE, DI, CCD, CDA, CAL	[18] [5] [10]

Tab. 4.3: Odporúčania pre filtrovanie prevádzky

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	IP spoofing	Špecifikácia ACL na zakázanie a logovanie privátnych a špeciálnych IP adries z RFC 1918, RFC 3330, RFC 6890, RFC 8190	C	C	CE, CCD, CAL	[5] [8] [10]
2	IP spoofing	Špecifikácia ACL na zakázanie a logovanie špeciálnych IPv6 adries z RFC 6890, RFC 8190, RFC 5156	C	C	CE, CCD, CAL	[5] [8] [10]
3	IPv6 Next Header, IPv6 Fragmentation útok	ACL blokujúce nerozpoznateľné rozšírené hlavičky	C	C	A	[16] [15]
4	DOS útok alebo pokus o prístup k tomu, čo nie je povolené	Logovanie pravidiel zahodenia paketov v ACL	M	M	A	[45]
5	Pakety budú spracovávané v CPU, ktoré môže byť preťažené a môže byť zmenené smerovanie na obídenie bezpečnostnej kontroly	Zahadzovanie IPv4 paketov s rozšírenou hlavičkou (IP Options filtering)	C	C	CE, DI, CCD, CDA, CAL	[8]
6	Komplexné bezpečnostné hrozby a narušenie bezpečnosti	Nastavenie IDS/IPS ak to zariadenie podporuje	C	H	CE, CCD, CAL	[46]

Tab. 4.4: Odporúčania pri vysokom zaťažení

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Nízky stav voľnej pamäte	Nastavenie notifikácie pri dochádzaní pamäte	M	M	A	[8] [21]
2	Logovacie správy nemôžu byť zaznamenané kvôli nedostatku pamäte	Rezervovanie pamäte pre kritické notifikácie pri nedostatku pamäte	M	H	A	[8] [21]
3	Vysoké zaťaženie CPU	Nastavenie notifikácie pri vysokom zaťažení CPU	M	M	A	[8] [21]
4	Vysoké zaťaženie zariadenia spôsobilo nemožnosť prihlásenia k nemu	Rezervovanie pamäte pre protokoly na manažment zariadení pri nedostatku pamäte	M	H	A	[8]

Tab. 4.5: Odporúčania na zamedzenie mapovania siete

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Skenovanie a zistenie informácií o sieti za pomoci protokolu CDP a využitie bezpečnostných chýb	Zakázanie protokolu CDP	M	C	A	[10] [20]
2	Skenovanie a zistenie informácií o sieti za pomoci protokolu LLDP a využitie bezpečnostných chýb	Zakázanie protokolu LLDP	M	C	A	[2]
3	Proxy ARP môže viesť k obídeniu PVLAN a rozširuje broadcast doménu	Vypnutie Proxy ARP	C	C	CE, DI, CCD, CDA, CAL	[10] [20]
4	Útočník môže zistiť, že IP adresa, na ktorú skúšal ping je nesprávna	Vypnutie správ ICMP Unreachable	D	H	CE, DI, CCD, CDA, CAL	[8]
5	Útočník môže zistiť masku podsiete pomocou ICMP Mask reply	Vypnutie správ ICMP Mask reply	D	H	CE, DI, CCD, CDA, CAL	[45]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
6	Umožňuje DOS Smurf útok, mapovanie siete pomocou ping na broadcast adresu vzdialenej siete	Vypnutie ICMP Echo správ na broadcast adresu, vypnutie Directed broadcasts	D	C	CE, DI, CCD, CDA, CAL	[20] [45]
7	Útočník môže zistiť smerovacie informácie alebo vyťažiť CPU	Vypnutie správ ICMP Redirects	D	H	CE, DI, CCD, CDA, CAL	[8]
8	Mapovanie siete pomocou pingu na multicast adresu všetkých uzlov a MLD/IGMP Query Overload a Smurf útok	ACL blokujúce ICMP Echo request na multicast adresu všetkých uzlov a MLD/IGMP Query na prístupových portoch	C	M	DI, CDA, AC	[33] [34]

Tab. 4.6: Odporúčania na identifikáciu zariadení a nastavení

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Nemožná identifikácia zariadenia	Vytvoriť hostname	M	L	A	[10]
2	Nemožnosť vzdialeného prístupu	Vytvoriť doménové meno	M	L	A	[10]
3	Nemožnosť identifikovať pravidlo v ACL	Popis každého pravidla v ACL pre lepšiu identifikáciu	M	L	A	[8]
4	Nemožnosť identifikovať rolu rozhrania	Vytvoriť popis každého rozhrania	M	L	A	[7]
5	Nemožnosť identifikácie účelu VLAN	Pridanie mena k VLAN	C	L	DI, CDA, AC, CAL	[7]

Tab. 4.7: Odporúčania k protokolu NTP

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Nekonzistencia časov v logoch a problém pričlenenia logov k relevantným incidentom	Nastavenie NTP serveru pre aktuálny čas v logoch	M	H	A	[5] [8] [10]
2	Pripojenie servera s rovnakou IP adresou, ale falošným časom	Nastavenie NTP autentifikácie	M	H	A	[5] [8] [10]
3	NTP zdrojové rozhranie nie je rovnaké pri každom reštarte	Definovanie loopback zdrojového rozhrania pre NTP	M	M	A	[5] [8] [10]
4	Väčšia bezpečnosť (pub/priv key) NTP a podpora IPv6	Použitie NTP verzie 4	M	M	A	[23]
5	Falošný čas od podvrhnutého NTP zdroja	Nastavenie NTP peer s inými sieťovými zariadeniami na krížovú validáciu času a záložný zdroj času	M	M	A	[45]

Tab. 4.8: Odporúčania pre protokol SNMP

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Odpočúvanie SNMP verzie 1 a 2c	Použitie SNMP verzie 3 pokiaľ je SNMP používané	M	C	A	[10] [20]
2	Modifikovanie konfigurácie pomocou SNMP	Obmedzenie SNMP iba na čítanie	M	C	A	[10] [20] [45]
3	Neoprávnený prístup k SNMP informáciám	Obmedzenie SNMP iba pre vybrané IP adresy	M	H	A	[10] [20]
4	Administrátor nemá povedomie o problémoch na zariadení	Povolenie asynchrónnych správ SNMP TRAP	M	M	A	[10] [20] [45]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
5	Odpočúvanie SNMP sedenia z dôvodu slabého šifrovania a hashovacej funkcie	Vytvorenie SNMP verzie 3 užívateľa s minimálnym šifrovaním AES 128 bit a hashovacou funkciou SHA	M	C	A	[11] [10] [45]
6	Sťažená identifikácia SNMP správ z rôznych IP	Definovanie lokácie SNMP serveru	M	L	A	[48]
7	SNMP zdrojové rozhranie nie je rovnaké pri každom reštarte	Definovanie loopback zdrojového rozhrania pre SNMP	M	M	A	[10]
8	Zmeny názvov rozhraní medzi reštartami a nemožnosť monitorovania pomocou SNMP	SNMP statické nemenné meno rozhrania aj po reštarte zariadenia	M	H	A	[48]

Tab. 4.9: Odporúčania pre protokol Syslog

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Administrátor nemá povedomie o problémoch na zariadení	Povolenie logovania protokolom SYSLOG a špecifikovanie IP adresy SYSLOG serveru	M	H	A	[10] [20]
2	Neprijímanie všetkých dôležitých incidentov na zariadení z protokolu SYSLOG	Špecifikovanie dôležitosti oznámení SYSLOG na INFORMATIONAL	M	M	A	[10]
3	SYSLOG zdrojové rozhranie nie je rovnaké pri každom reštarte	Definovanie loopback zdrojového rozhrania pre SYSLOG	M	M	A	[8] [10]
4	Nedostatočné a neštandardné formáty času pri logovacích správach	Definovanie formátu času pre logovacie a ladiace výstupy	M	M	A	[10] [20]
5	Administrátor nevidí dôležité incidenty pri prihlásení a konfigurovaní cez konzolu	Vypisovanie SYSLOG správ CRITICAL a dôležitejších do terminálu	M	M	A	[8] [10]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
6	Malá vyrovnávacia pamäť pre SYSLOG je dôvodom zahadzovanie správ	Definovanie veľkosti SYSLOG buffera dôležitosti oznámení na INFORMATIONAL	M	H	A	[8]
7	Neprístupný SYSLOG server spôsobuje zahadzovanie dôležitých syslog správ	Definovanie dočasného úložiska SYSLOG správ v prípade nedostupnosti servera	M	H	A	[8]
8	Problém identifikácie SYSLOG správ s rovnakou časovou značkou	Pridanie sekvenčného čísla ku každej syslog správe	M	L	A	[45]

Tab. 4.10: Odporúčania pre First Hop Security

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	MAC Spoofing, MAC Flooding	Definovanie maximálne 1 MAC adresy na port, priradenie MAC adresy na port	C	C	DI, CDA, AC	[7]
2	MAC Spoofing, MAC Flooding	Nastavenie režimu narušenia, ktorý vypne port alebo informuje správcu o pripojení nepovoleného zariadenia	C	H	DI, CDA, AC	[7]
3	Využívanie siete nepovolenými používateľmi	Zapnutie 802.1x	C	H	DI, CDA, AC	[3] [36] [37]
4	Útok hrubou silou hádaním prístupových údajov pre 802.1x	Limitovanie maximálneho počtu neúspešných pokusov o autentifikáciu 802.1x	C	H	DI, CDA, AC	[3] [36] [37]
5	DHCP spoofing	DHCP snooping, IPv6 Snooping, DHCPv6 Guard	C	C	DI, CDA, AC	[3] [8] [35]
6	Príliš veľa DHCP paketov, zaplavenie DHCP paketmi	Obmedziť počet DHCP paketov na nedôveryhodných rozhraniach	C	M	DI, CDA, AC	[3] [8] [35]
7	ARP Spoofing	Dynamic ARP Inspection	C	C	DI, CDA, AC	[2]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
8	IP spoofing	IPv4/IPv6 Source Guard	C	C	DI, CDA, AC	[8] [35]
9	IPv6 ND Spoofing	IPv6 ND Inspection	C	C	DI, CDA, AC	[29] [28] [35]
10	Rogue RA, RA Flood, RouterLifeTime=0	RA Guard	C	C	DI, CDA, AC	[29] [28] [35]
11	Mobilné zariadenia pripojené bezdrôtovo spotrebovávajú veľa energie kvôli častým RA správam	RA Throttling	C	L	DI, CDA, AC	[33] [40]
12	Vyčerpanie cache susedov	Statický záznam pre kritické zariadenia (servery) spájajúce IP a MAC adresu a VLAN	C	C	DI, CDA, AC	[31] [32]
13	Vyčerpanie cache susedov	Na zabránenie vzdialeného útoku na cache susedov cez internet je potreba nastaviť ACL, kde povoľujeme iba komunikáciu s cieľovými IPv6 adresami, ktoré sa nachádzajú v našej sieti	C	C	CE, CCD, CAL	[31] [32]
14	Vyčerpanie cache susedov	IP destination Guard (First Hop Security)	C	C	DI, CDA, AC	[31] [32]
15	Vyčerpanie cache susedov	Limitovanie času IPv6 adresy v cache susedov	C	C	DI, CDA, AC	[31] [32]

Tab. 4.11: Odporúčania pre Spanning Tree Protokol

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Rogue root bridge	Rogue root bridge protection (root guard)	C	C	DI, CDA, AC	[3]
2	Pripojenie prepínaču na koncový prístupový port	BPDU protection (BPDU guard)	C	C	DI, CDA, AC	[3]
3	Rýchlosť konvergencie	Prístupové porty by sa nemali podieľať na STP procese	C	H	DI, CDA, AC	[3]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
4	Jednosmerná komunikácia medzi prepínačmi môže viesť k topológii so slučkami	Špeciálne konfigurácie zaisťujúce bezslučkovú topológiu pomocou STP keď nastane jednosmerná komunikácia (Loop Guard)	C	C	DI, CDA, AC	[50]

Tab. 4.12: Odporúčania pre VLAN

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Špeciálna VLAN pre manažment na obmedzenie prístupu iba pre administrátorov	Vytvorenie separátnej VLAN pre manažment	C	M	DI, CDA, AC	[7]
2	Útočníkovi s fyzickým prístupom k portu môže byť pridelený prístup do časti siete, ktorá zodpovedá príslušnej VLAN	Vytvorenie špeciálnej black hole VLAN pre nevyužité porty	C	C	DI, CDA, AC	[21]
3	Predvolenej VLAN je povolené prepnutie na akýkoľvek port, VLAN hopping, double tagging	Odobrať všetky porty z predvolenej VLAN	C	C	DI, CDA, AC	[21]
4	Predvolenej VLAN je povolené byť prepnutá na akýkoľvek port, VLAN hopping, double tagging	Vytvorenie natívnej VLAN rozdielnej ako predvolená, priradenie k trunk portu a povolenie iba potrebných portov	C	C	DI, CDA, AC	[21]
5	DTP útok, Switch spoofing útok	Vypnutie dynamického trunkovacieho protokolu a explicitne určiť porty ako prístupové a trunk	C	C	DI, CDA, AC	[21]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
6	Nový prepínač s vyšším číslom revízie, ale s nesprávnou VLAN databázou môže šíriť falošné VLAN identifikátory a spôsobiť nefunkčnosť siete, veľa možných VTP útokov kvôli zraniteľnostiam	Vypnutie MVRP, MRP, GARP, VTP po úspešnej propagácií VLAN	C	C	DI, CDA, AC	[3]
7	VTP musí byť používané	Uprednostniť VTP verzie 3, špecifikovať skryté heslo a zapnúť VTP prunning pokiaľ musí byť VTP zapnuté	C	C	DI, CDA, AC	[3]
8	Vysoké zaťaženie linky	Poslanie notifikácie pri prekročení prahovej hodnoty zaťaženia linky	C	M	A	[21]

Tab. 4.13: Ostatné nezatriedené odporúčania

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
1	Zlyhanie zariadenia alebo linky môže viest k nefunkčnosti siete	Povolenie FHRP s autentifikáciou a aktuálnou verziou	C	M	CE, CCD, CAL	[7]
2	Nepoužívané, staré a nezabezpečené služby môžu byť použité na škodlivé účely	Vypnutie nepoužívaných služieb z bezpečnostných dôvodov a na šetrenie CPU a pamäte	*1	H	*1	[38]
3	Odpočúvanie komunikácie cez nezabezpečené tunely	Vypnúť tunely ktoré nie sú zabezpečené alebo zabezpečiť tunely	D	C	CE, DI, CCD, CDA, CAL	[10] [46]
4	Môže byť zneužité odpočúvanie pokiaľ sa používa a monitorovanie prevádzky kvôli legislatívnym potrebám (zrkadlenie portov, záznam tokov)	Monitorovanie výkonnosti siete a zber sieťového prenosu kvôli legislatívnym potrebám	C	N	A	[8]

Riadok č.	Útok / Problém	Mitigácia / Nastavenie	Plane	Severity	Facility layer	Zdroj
5	Útočník s fyzickým prístupom k zariadeniu alebo portu môže odpočúvať alebo posielať škodlivý obsah	Explicitne zakázať nepoužívané porty	D	C	A	[20] [5]
6	Zdrojové rozhranie pre management a control protokoly	Vytvoriť Loopback rozhranie s IP adresou	M/C	M	A	[8] [10]
7	Pretečenie pamäte	Povoliť mechanizmy na detekciu pretečenia pamäte	M	M	A	[8]
8	Načítanie škodlivej konfigurácie zo siete počas bootovania	Vypnutie načítania operačného systému alebo konfigurácie zo siete, pokiaľ to nie je nutné	M	M	A	[45]
9	Odpočúvanie konfigurácií zariadení pri zálohe	Zapnutie zabezpečenej zálohy na server (SFTP, SCP)	M	H	A	[8]
10	Vymazanie konfigurácie	Zapnutie ochrany pred výmazom konfigurácie	M	H	A	[2]
11	Možnosť urobiť rozdiel zmien konfigurácií a jej návrat	Periodické zálohovanie konfigurácie a logovanie jej zmien	M	M	A	[2] [8]
12	Nemožnosť prihlásenia pri zaseknutom TCP spojení	Terminovanie zaseknutého TCP spojenia	M	M	A	[8]
13	Možnosť prečítať heslá z uniknutých konfigurácií	Zašifrovanie hesiel v otvorenej podobe	M	C	A	[10]

[1]Záleží na výrobcovi, operačnom systéme a verzii

5 Implementácia

V nasledujúcej kapitole budú opísané použité technológie na implementáciu práce, rozobraté dôležité časti a princíp ich fungovania, ako aj konfiguračné súbory zodpovedné za nájdenie a nápravu nedostatkov. Vo výpisoch kódu ako aj zdrojových súboroch (komentároch) sa môžu objaviť pasáže písane v anglickom jazyku. Napriek faktu, že je celá práca písaná v slovenčine, tak pre zdrojové súbory programu je angličtina vhodnejšia z hľadiska univerzálnosti a tiež kvôli predpokladanému budúcemu využitiu programu ďalšími osobami, keďže program bude verejne dostupný. Výpisy kódu v nasledujúcich kapitolách sú skrátené a iba ilustračné.

5.1 Použité technológie

5.1.1 Python

Python [53] je objektovo orientovaný, interpretovaný programovací jazyk vytvorený holanďanom Guidom van Rossom. Radí sa k vysokoúrovňovým programovacím jazykom a umožňuje automatickú správu pamäte, teda programátor nie je nútený explicitne potrebnú pamäť alokovať a uvoľňovať Jeho výhodou je práve fakt, že je interpretovaný a teda programy napísané v ňom nemusia byť preložené pre danú platformu, ale postačuje spustenie zdrojového kódu pomocou interpretu nainštalovaného na danom systéme. Interpret jazyka Python je dostupný pre Microsoft Windows, GNU/Linux, macOS a mnoho ďalších vstavaných a exotickejších systémov. Syntax jazyku Python je založená na syntaxi jazyku C, no zdrojový kód je čitateľnejší a na vymedzenie funkčných blokov využíva odsadenie, vďaka čomu je kód čitateľnejší. V súčasnosti sa využíva syntax a interpret dvoch verzií, a to verzie 2.x a 3.x, ktoré sú vzájomne nekompatibilné, z dôvodu rozdielnych syntaktických konštrukcií. Prvá zo zmienených verzií však čoskoro prestane byť podporovaná. Z tohto dôvodu je vhodnejšie použiť na novo vyvíjané programy verziu 3.x.

Práve pre vyššie zmienené výhody, a to najmä dobrú čitateľnosť kódu, rozšíriteľnosť medzi programátormi, ako aj spustenie na rôznych platformách bol Python zvolený za programovací jazyk pre túto diplomovú prácu.

5.1.2 YAML

YAML [54] je jazyk na serializáciu dát vo forme veľmi dobre čitateľnej pre človeka. Bol inšpirovaný konceptami a syntaxou jazykov C a Python. Dovoľuje definovať primitíva z týchto programovacích jazykov ako zoznamy, asociatívne polia, reťazce a zároveň v jednom súbore dovoľuje definovať viac dokumentov.

Pre konfiguračné súbory na túto diplomovú prácu boli spočiatku uvažované tri metódy. Prvou možnosťou na serializáciu dát bol jazyk XML, tento formát však nie je tak dobre čitateľný pre človeka a je tu väčšia pravdepodobnosť zanesenia chýb z dôvodu nutnosti uzatvárať značky. Druhou možnosťou bolo využitie syntaxe jazyka JSON, no problémom je, že nepodporuje vkladanie komentárov, preto nie je vhodný na konfiguračné súbory, ale skôr na strojovú serializáciu. Poslednou možnosťou bolo vytvorenie vlastnej syntaxe a vlastného analyzátoru, to je však pre potreby diplomovej práce zbytočné a hotové riešenie v podobe YAML je dostatočné a eliminuje všetky nedostatky vyššie zmienených jazykov určených na serializáciu dát. Použitý jazyk Python naviac disponuje viacerými knižnicami na prácu s jazykom YAML.

5.1.3 Regulárne výrazy

Regulárnym výrazom sa rozumie sekvencia znakov, ktorá definuje určitý vzor [55]. Regulárne výrazy sa používajú na vyhľadávanie alebo výmenu reťazcov v texte pričom na to využívajú znaky s vopred definovanou sémantikou. V diplomovej práci budú použité na vyhľadávanie prítomnosti alebo absencii nastavení v konfigurácií zariadení.

5.2 Konfiguračné súbory

Konfiguračné súbory YAML sú neoddeliteľnou súčasťou samotného programu a slúžia súbežne na viacero funkcionalít programu. Prvým z nich je `device_info.yaml` 5.1, ktorý obsahuje základné údaje o testovanom zariadení, pričom pre každé testované zariadenie je separátny súbor. Druhým typom je súbor uchovávajúci informácie o module, obsahujúci regulárne výrazy na hľadanie v nastaveniach zariadenia, informácie na zjednanie nápravy a mnohé ďalšie režijné informácie. Vo výpise kódu 5.2 je modul `01_02_aaa_server.yaml` zodpovedný za definovanie AAA serveru. Ďalším YAML súborom v poradí je `modules_by_facility_layer.yaml` 5.3, definujúci, ktoré moduly budú spustené na zariadení podľa vrstvy hierarchického modelu, na ktorom zariadenie operuje. Posledný dôležitý YAML súbor – `own_variables.yaml` 5.4 obsahuje potrebné premenné na prípadnú zjednávanú nápravu pre danú topológiu. Podrobne budú jednotlivé súbory a ich dôležité položky rozpísané v nasledujúcich podkapitolách.

5.2.1 Ukladanie informácií o zariadení

Veľmi dôležitou súčasťou analýzy je zistenie informácií o zariadení. Na uchovanie týchto informácii slúži súbor `device_info.yaml` 5.1. Okrem základých informá-

cií ako mena zariadenia (`hostname`), mena súboru s exportovaným nastavením zariadenia (`config`) a verzie operačného systému (`version`) obsahuje aj ďalšie informácie. Pre správnu analýzu je nutné vedieť, či je na zariadení podpora pre IPv4 a IPv6 (`l3_protocols`) a či sú oba protokoly povolené. Premenné `vendor` a `os` zasa určujú, o akého výrobcu sa jedná a aký operačný systém beží na zariadení. Toto sú kľúčové položky, nakoľko ich kombinácia určuje cestu ku všetkým YAML šablóna pre daného výrobcu a operačný systém. Tieto položky sa vypĺňujú na základe argumentov pri spúšťaní prvotného príkazu: `netsec.py analyze --workspace <workspace> --vendor <vendor> --os <os>`. Položky `facility` a `facility_layer` sa generujú automaticky na základe analýzy exportovanej konfigurácie zariadenia. Druhá zmienená položka je podrobne rozobraná v kapitole 5.3.2. Dvojica premenných `include_modules` a `exclude_modules` umožňujú pridávať alebo odoberať spúšťané moduly, ktoré by boli spustené na základe definície v súbore `modules_by_facility_layer.yaml`, a tým personalizovať skenovanie na špecifiká danej topológie a politike nastavení pre danú firmu. Súčasťou informácií o zariadení je aj výčet jednotlivých rozhraní `interfaces` a pridelenie určitých nájdených stavov a funkcií, ktoré sú potrebné pri jednotlivých moduloch hľadajúcich nedostatky. Nutné je tiež vedieť, aké funkcionality `enabled_functions` sú na zariadení spustené. Nemá totiž zmysel testovať bezpečnostné nastavenia pre EIGRP, pokiaľ EIGRP nie je na zariadení používané. Generovalo by to zbytočne falošne pozitívne oznámenia. Položka `input_config_hash` obsahuje SHA1 hash exportovanej konfigurácie, ku ktorej robíme analýzu. Dôležitá je nielen na prípadné spätné zistenie, ku ktorej konfigurácií analýza patrí, ale aj z dôvodu výslednej správy o nedostatkoch. Pretože ak by sme my alebo niekto iný pozmenil vstupnú exportovanú konfiguráciu zo zariadenia pre analýzu, ktorá už prebehla a máme výslednú správu, mohlo by to spôsobiť rozpor medzi nájdenými nedostatkami a stavom v pozmenenej konfigurácii. Takto máme dôkaz o tom, že niekto konfiguráciu pozmenil. Samozrejme hash novej konfigurácie sa dá priložiť do YAML súboru, takže ochrana nie je ideálna. Pokiaľ však vygenerujeme správu o analýze hneď po samotnej analýze nedostatkov a túto správu vytlačíme, prípadne uložíme na iné miesto, tak máme nesporný dôkaz o manipulácií so vstupnými údajmi. Pozmenenie vstupu sa však dá vykonať iba ak by mal záškodník prístup k pôvodným vstupom a chcel by spochybniť výsledky analýzy. Týmto záškodníkom môže byť aj samotný administrátor, ktorý by chcel podvrhnúť výsledky analýzy, to by bol ale veľmi extrémny prípad. Podobným spôsobom funguje aj posledná premenná `fix_hash` uchovávajúca hash súboru s generovanou nápravou.

Výpis 5.1: Konfiguračný súbor `device_info.yaml`, ktorý popisuje základné informácie o jednom konkrétnom zariadení

```yaml
---
hostname: "SW1_ACCESS"
config: "SW1_ACCESS_conf.txt"
version: "15.4"
l3_protocols:
  - "ipv4"
  - "ipv6"
vendor: "cisco"
os: "ios"
facility: "l3sw"
facility_layer: "collapsed_distribution_access"
exclude_modules: []
include_modules: []
interfaces:
  Port-channel1:
  - "trunk"
  - "noip"
  - "port-channel"
  Ethernet0/0:
  - "trunk"
  - "noip"
  - "channel-group1"
  ...
enabled_functions:
  - "rip"
  - "eigrp"
input_config_hash: "2ad9b080449de7a86e88b56fb2993b7f3ba4308e"
fix_hash: ""
```

5.2.2 Popis modulov

Základným stavebným pilierom celého programu sú moduly alebo inak povedané konfiguračné súbory YAML. Obsahujú všetky dôležité náležitosti potrebné k nájdeniu doporučeného nastavenia, respektíve jeho neprítomnosti na zariadení, dáta k zjednaniu nápravy, ako aj notifikačné informácie pre správu o analýze. Všetky komentáre, ktoré sú prítomné v YAML moduloch budú zobrazené v správe o analýze.

Pomenovanie modulov, napríklad `01_02_aaa_server.yaml` 5.2 je charakteristické dvojicou čísiel. Prvá dvojica je rovnaká pre viacero modulov a identifikuje oblasť, o ktorú sa modul stará, napríklad nastavenia AAA. Druhá dvojica je unikátna vrámci oblasti AAA, pričom v ostatných oblastiach môže byť použité rovnaké číslo. Toto číslovanie je povinné, pri pridaní nového modulu do oblasti je potreba zachovať číslovanie a zároveň druhá dvojica čísiel sa nemôže v danej oblasti opakovať. Číslovanie bolo použité z dôvodu, aby sa moduly spúšťali v želanom poradí, nebolo nutné rekurzívne spúšťanie, kde bolo nutné definovať viac náväzností ako v súčasnosti. Naviac to umožňuje pri generovaní správy s nedostatkami vypisovať spolu podobné moduly z jednej oblasti.

Výpis 5.2: Konfiguračný súbor `01_02_aaa_server.yaml`, ktorý popisuje základné informácie o jednom konkrétnom zariadení

```
---
type: "m"
facility_type: []
check_if_l3_protocol: []
check_if_function: []
run_after_module: "01_01_aaa_new_model.yaml"
run_after_module_match_status: "none"
applicable_to_interface_type: []
non_applicable_to_interface_type: []
cannot_determine_search_or_fix: "false"
cannot_determine_search_or_fix_comment: ""
eliminated: "false"
name_cmd_general: "AAA server set"
name_of_area: "Authentication, Authorization, Accounting"
default_cmd_general_severity: "high"
user_cmd_general_severity: "none"
regex_cmd:
- '^(radius|tacacs) server (.+)$(?:.*\r?\n(?!\!))+?^.*address
    ipv[46].+$(?:\r?\n^.*key.+$)?(?:.*\r?\n)*?(?=\!)'
```

```
- '^.*(radius|tacacs)-server host .*$(\r?\n^.*(radius|tacacs)
   -server key.*$)?'
- '^.*(radius-server|tacacs-server) host .* key.+$'
regex_context: ""
regex_cmd_occurrence: "occurrence"
cmd_match_status: "not run"
general_comment: ""
mark_module_as:
- NOMODULE: "none"
- 01_03_aaa_group.yaml: "matched by equivalent"
- 01_03_aaa_group.yaml: "matched by equivalent"
matched_values: []
public_vars:
- aaa_type: "group(1)"
- aaa_server_name: "regex_cmd[0].group(2)"
- aaa_ip: ""
- aaa_key: ""
secret_vars: []
eliminate_all_matched: "false"
eliminate_prefix: ""
fix_cmd:
- '$aaa_type server $aaa_server_name'
- 'address ipv4 $aaa_ip'
- 'address ipv6 $aaa_ip'
- 'key $aaa_key'
fix_to_apply: ""
fix_cmd_notice: ""
fix_cmd_ignore: "false"
fix_cmd_ignore_comment: ""
fix_cmd_false_positive: "false"
fix_cmd_false_positive_comment: ""
affected_ports: []
affected_context: []
explicit_ignored_ports: []
explicit_ignored_ports_comment: ""
```

Premenné a ich význam:

- `type` – môže nadobudnúť štyri hodnoty, ktoré kopírujú rozdelenie príkazov z kapitoly 4.2, preto ich význam a motivácia už nebudú ďalej rozoberané.
- `facility_type` – aj keď definujeme moduly, ktoré sa majú spustiť na základe príslušnosti zariadenia vo vrstve hierarchického modelu, môžu nastať situácie,

kedy to nestačí. Takýmto príkladom je distribučná vrstva. Na tejto vrstve môžu byť použité buď L3 prepínače, alebo smerovače. Rozdiel je badateľný napríklad v module, ktorý overuje nastavenia natívnej VLAN trunk portov. Pokiaľ by bolo zariadenie smerovač, tak tento modul nemá zmysel spúšťať, pretože by generoval iba falošne pozitívny nález.

- `check_if_l3_protocol` – definuje, že modul bude spustený iba ak je na zariadení povolený určitý L3 protokol napríklad IPv6. V prípade prázdnej premennej na tom aký je protokol prítomný a nastavený na zariadení nezáleží, modul bude spustený vždy.
- `check_if_function` – definované funkcie ako napríklad OSPF, ktoré musia byť prítomné na zariadení, ak má byť spustená analýza. Je to z dôvodu eliminovania falošne pozitívnych správ, pretože nemá zmysel hľadať nedostatky v OSPF, ak vôbec nie je na zariadení nastavené. V prípade prázdnej premennej, nezáleží na tom, aké funkcie sú prítomné a nastavené na zariadení, modul bude spustený vždy.
- `run_after_module` – špecifikovanie modulu, ktorý musí byť spustený pred aktuálnym. Tejto premennej nemusí byť vždy priradená hodnota.
- `run_after_module_match_status` – zväzujúca informácia s predchádzajúcou položkou. Pri definovaní `none` stačí ak daný predchádzajúci modul bol spustený. Ak bude hodnota `error`, príde k spusteniu aktuálneho modulu iba v prípade neúspechu predchádzajúceho modulu. Analogicky funguje hodnota `successful`, kedy musí predchádzajúci modul skončiť úspechom, aby sa aktuálny spustil.
- `applicable_to_interface_type` – v prípade ak ide o modul, ktorý kontroluje rozhrania, teda `type: "i"`, tak sa môže špecifikovať zoznam, na základe ktorého sa bude testovať iba prítomnosť nastavenia na rozhraní, ktoré má určité nastavenie, napríklad na prístupových portoch `access`. V prípade viac ako jednej hodnoty v tomto zozname sa použije logická funkcia OR, a prítomnosť nastavenia sa bude hľadať na všetkých rozhraniach, ktoré spĺňajú aspoň jeden z definovaných prvkov zoznamu. V prípade prázdneho zoznamu sa bude nastavenie hľadať na všetkých rozhraniach bez výnimky.
- `non_applicable_to_interface_type` – premenná, ktorá funguje podobne ako predchádzajúca, akurát môže bližšie špecifikovať rozhranie, teda aj napriek pozitívnemu výskytu a zhode v predchádzajúcom parametre môžu nastať kombinácie, kde analýza modulom nie je na mieste. Napríklad by rozhranie malo atribút `access` a daný modul je aplikovateľný na takéto rozhrania, tak môžeme špecifikovať, že je aplikovateľný ak je `access` a zároveň nie je `shutdown`.
- `cannot_determine_search_or_fix` – môže nastať prípad, že nemáme špecifikované všetky premenné v `own_variables.yaml`, prípadne pri hľadaní nedos-

tatkov program nebol schopný premenné vyplniť. V takomto prípade sa táto premenná označí ako `True`, čo znamená, že administrátor musí spozornieť a prečítať si komentár k tomuto nálezu programu. Taktiež môže nastať stav, že nie je možné nájsť daný kontext špecifikovaný v `regex_context` alebo v prípade nastavenia na rozhraní, nebolo nájdené rozhranie s daným nastavením napríklad `access`, na ktorého typ modul hľadá nedostatok.

- `cannot_determine_search_or_fix_comment` – dodatočný komentár k vysvetleniu nadobudnutia hodnoty `true` v predchádzajúcej premennej.
- `eliminated` – znižuje počet falošne pozitívnych správ, program označí premennú ako `true` v prípade, že na skenovanom zariadení nie je funkcia definovaná v `check_if_l3_protocol` alebo `check_if_function`.
- `name_cmd_general` – názov modulu, ktorý bude vidieť v záverečnej správe.
- `name_of_area` – oblasť pod ktorú spadá daný modul, obsah premennej bude viditeľný v záverečnej správe.
- `default_cmd_general_severity` – predvolená hodnota závažnosti nastavenia respektíve jeho nedostatku.
- `user_cmd_general_severity` – špecifická hodnota závažnosti nastavenia respektíve jeho nedostatku zmenená administrátorom, ktorá bola zistená analýzou rizík.
- `regex_cmd` – regulárny výraz potrebný na nájdenie nastavenia. V prípade viac ako jedného reťazca v zozname sa použije logická funkcia OR pričom pri neúspešnom hľadaní prvého výrazu sa skúša ďalší v poradí. Hľadanie sa rozumie ako úspešné, pokiaľ aspoň jeden z regulárnych výrazov nájde zhodu v konfigurácií zariadenia. Všetky regulárne výrazy sú si rovnocenné a hľadajú rovnaké nastavenie, ale s rozličnou syntaxou.
- `regex_context` – kontext pod ktorým sa hľadá regulárny vyraz z `regex_cmd`. V tejto premennej sa nachádza tiež regulárny výraz, využitá je iba v prípade bodu 4 kapitoly 4.2.
- `regex_cmd_occurrence` – nie vždy je žiadúce aby hľadaný regulárny výraz našiel zhodu, pretože niektoré nastavenia zariadení spôsobujú bezpečnostnú hrozbu. Môžu byť prednastavené už od výroby alebo aplikované administrátormi. Z tohto dôvodu táto premenná pri nadobudnutej hodnote `occurence` definuje, že hľadaný regulárny výraz musí nájsť dané nastavenie a musí byť v zariadení prítomné. Naopak `non-occurence` znamená, že analýza modulom je úspešná, ak daný regulárny výraz nenájde žiadnu zhodu.
- `cmd_match_status` – definuje úspešnosť `successful` alebo neúspešnosť `error` pri hľadaní regulárneho výrazu. Taktiež zo začiatku ešte nespustený modul označuje ako `not run`. V prípade, že existuje ešte nejaký modul, ktorého nastavenie implikuje súčasný modul, tak sa v tejto premennej ocitne reťazec

`matched by equivalent`.

- `mark_module_as` – ako už bolo spomenuté v predchádzajúcom bode, tak pri úspešnom nájdení nastavenia respektíve pri explicitne vyžiadanej neprítomnosti nastavenia môžeme označiť iný modul povolenými hodnotami z predchádzajúceho bodu. Poradie v zozname je ekvivalentné s poradím z premennej `regex_cmd`. V prípade príkladu z výpisu nižšie to znamená, že pri úspešnom nájdení prvého regulárneho výrazu sa neoznačí žiaden modul. Pokiaľ sa však nenájde zhoda s prvým regulárnym výrazom testuje sa ďalší a pri jeho zhode sa označí modul `01_03_aaa_group.yaml` ako `matched by equivalent`.
- `general_comment` – všeobecný komentár, do ktorého sa ukladajú rôzne výstupy a upozornenia z programu.
- `matched_values` – všetky zhody, ktoré sa nájdu v nastavení zariadenia budú uložené do tejto premennej. Z hľadiska anonymizácie výstupnej správy je však možné nič do tejto premennej programom nezapisovať.
- `public_vars` – na zjednanie nápravy je často potrebné poznať premenné do niektorých príkazov, ktoré nastavujú zariadenie. Tie môžeme získať z úspešného nájdenia príkazu v konfigurácií, alebo zo súboru `own_variables.yaml`. Detailnejšie sa problémom zaoberá kapitola 5.2.4.
- `secret_vars` – nie je v súčasnej dobe využitá, počíta sa s premiestnením premenných na zjednanie nápravy z `public_vars`, ktoré by sa nemali uchovávať z bezpečnostných dôvodov, napríklad rôzne heslá, ktoré sú súčasťou príkazov.
- `eliminate_all_matched` – administrátor zvolí túto hodnotu `true` ak máme `regex_cmd_occurrence: non-occurence`, teda ak sa nájde v nastavení príkaz, ktorý by tam nemal figurovať.
- `eliminate_prefix` – súvisí z prechádzajúcou premennou a definuje reťazec, ktorý sa pridá k nájdenému nastaveniu, ktoré by sa v ideálnom prípade nemalo v konfigurácií zariadenia nachádzať a tým toto nastavenie eliminuje. Príkladom môže byť prefix `no` alebo `delete`.
- `fix_cmd` – príkaz zjednávajúci nápravu. Pokiaľ má príkaz viacej riadkov, každý riadok sa píše ako zvlášť položka zoznamu. Je to z dôvodu lepšej čitateľnosti a jednoduchšieho pridávania premenných, ktoré majú začiatočný symbol `$`. Platí, že sa môžu použiť premenné iba definované v danom module v premennej `public_vars`.
- `fix_to_apply` – generované programom, výsledné príkazy s vyplnenými premennými v jednom reťazci.
- `fix_cmd_notice` – Poznámka, ktorá môže byť pridaná a oznamuje prípadné problémy, ktoré by mohli vzniknúť aplikovaním nápravy.
- `fix_cmd_ignore` – administrátor môže zadať `true` pokiaľ nechce aby sa generovala náprava pre tento typ modulu.

- `fix_cmd_ignore_comment` – komentár a zdôvodnenie, prečo administrátor nechce povoliť automatické generovanie nápravy
- `fix_cmd_false_positive` – administrátor môže zadať `true` pokiaľ je nález falošne pozitívny.
- `fix_cmd_false_positive_comment` – komentár a zdôvodnenie, prečo ide o falošne pozitívny nález.
- `affected_ports` – výpis rozhraní, ktoré absentujú dané nastavenie. Táto premenná je programom vyplnená iba ak `type: "i"`.
- `affected_context` – výpis kontextov, ktoré absentujú dané nastavenie. Táto premenná je programom vyplnená iba ak `type: "c"`.
- `explicit_ignored_ports` – výpis rozhraní, ktoré sú explicitne vylúčené z analýzy.
- `explicit_ignored_ports_comment` – komentár a zdôvodnenie, prečo boli rozhrania vylúčené.

Zaujímavou časťou je označovanie iných modulov bez ich samotného spustenia pomocou reťazca `mark_module_as`. Dôležité je povedať, že takéto označovanie iných modulov je možné iba v prípade úspešnosti aktuálneho modulu. Pri nájdení chyby respektíve nedostatku to nie je možné. Iný modul je možné označiť vopred, teda ešte pred jeho samotným spustením, v tom prípade k jeho spusteniu ani nedôjde. Je však možné nejaký modul označiť aj spätne, pričom ak v označovanom module bola predtým nájdená chyba a vygenerovaná náprava, tak sa táto náprava eliminuje. Pri takomto spätnom a doprednom označovaní sa odporúča modul označiť ako `matched by equivalent`.

5.2.3 Súbor popisujúci vrstvy a moduly

Program má ako jednu z požiadaviek minimalizovať počet falošne pozitívnych hlásení. Jeden z prístupov, ktorý má tomuto zamedziť, je spúšťanie kontroly prítomnosti nastavení na základe umiestnenia zariadenie v hierarchickom modeli siete. Preto je nutné špecifikovať, ktoré moduly majú byť spustené na zariadeniach príslušnej vrstvy. Rozdelenie vrstiev je spomenuté v kapitole 4.3 a na základe tohto rozdelenia je vytvorený aj súbor `modules_by_facility_layer.yaml`. Následný výpis kódu 5.3 je len ilustračný a skrátený, kde bodky symbolizujú pokračovanie. Jeden YAML modul, môže byť definovaný na viacerých vrstvách. Špeciálnym prípadom je vrstva `all`, ktorá vlastne ani vrstvou nie je. Ide o zápis, ktorý odstraňuje redundanciu, teda moduly v nej definované by museli byť zadané povinne na každej vrstve, čo by spravilo súbor menej prehľadným a aj komplikovalo editáciu.

Výpis 5.3: Ukážka skráteného príkladu skráteného konfiguračného súboru `modules_by_facility_layer.yaml`, ktorý popisuje, aké moduly môžu byť spúšťané na jednotlivých vrstvách.

```
---
all:
- "01_01_aaa_new_model.yaml"
- "01_02_aaa_server.yaml"
...
- "02_20_session_timeout.yaml"
...
- "11_02_cdp_enable.yaml"
...
core:
..
- "14_01_filter_private_on_wan.yaml"
..
collapsed_all:
...
- "10_01_access_vlan_1.yaml"
...
collapsed_core_distribution:
...
- "10_01_access_vlan_1.yaml"
...
distribution:
...
- "10_01_access_vlan_1.yaml"
```

```
...
collapsed_distribution_access:
...
- "10_01_access_vlan_1.yaml"
...
access:
...
- "10_01_access_vlan_1.yaml"
...
```

5.2.4 Premenné na generovanie nápravy

Pre generovanie náprav je nutná znalosť nielen samotného príkazu, ktorý problém vyrieši, ale je často potreba doplniť do príkazov premenné ktoré sa môžu pre jednotlivé testované siete líšiť. Preto vznikol YAML súbor, ktorého čiastočná ukážka je vo výpise 5.4. Po spustení programu `netsec.py` s argumentom `analyze` môžeme vyplniť tento YAML súbor v príslušnom workspace. Je dobré vyplniť čo najviac premenných, ideálne všetky, pretože nikdy dopredu nevieme, ktorý hľadaný nedostatok bude potrebovať nápravu a informácie práve z tohto súboru. Pri nevyplnení niektorých premenných je administrátor notifikovaný touto skutočnosťou a môže prázdne premenné doplniť alebo ignorovať túto notifikáciu. Prioritne sa však premenné neberú z `own_variables.yaml`, ale k ich výberu dochádza až ako k poslednej možnosti. Prioritne sa berú informácie z modulov, ktorého jeden reprezentant je v kapitole 5.2.2 v premennej `public_vars`. Tieto premenné sa mohli nájsť pri úspešnom hľadaní nejakého predchádzajúceho modulu.

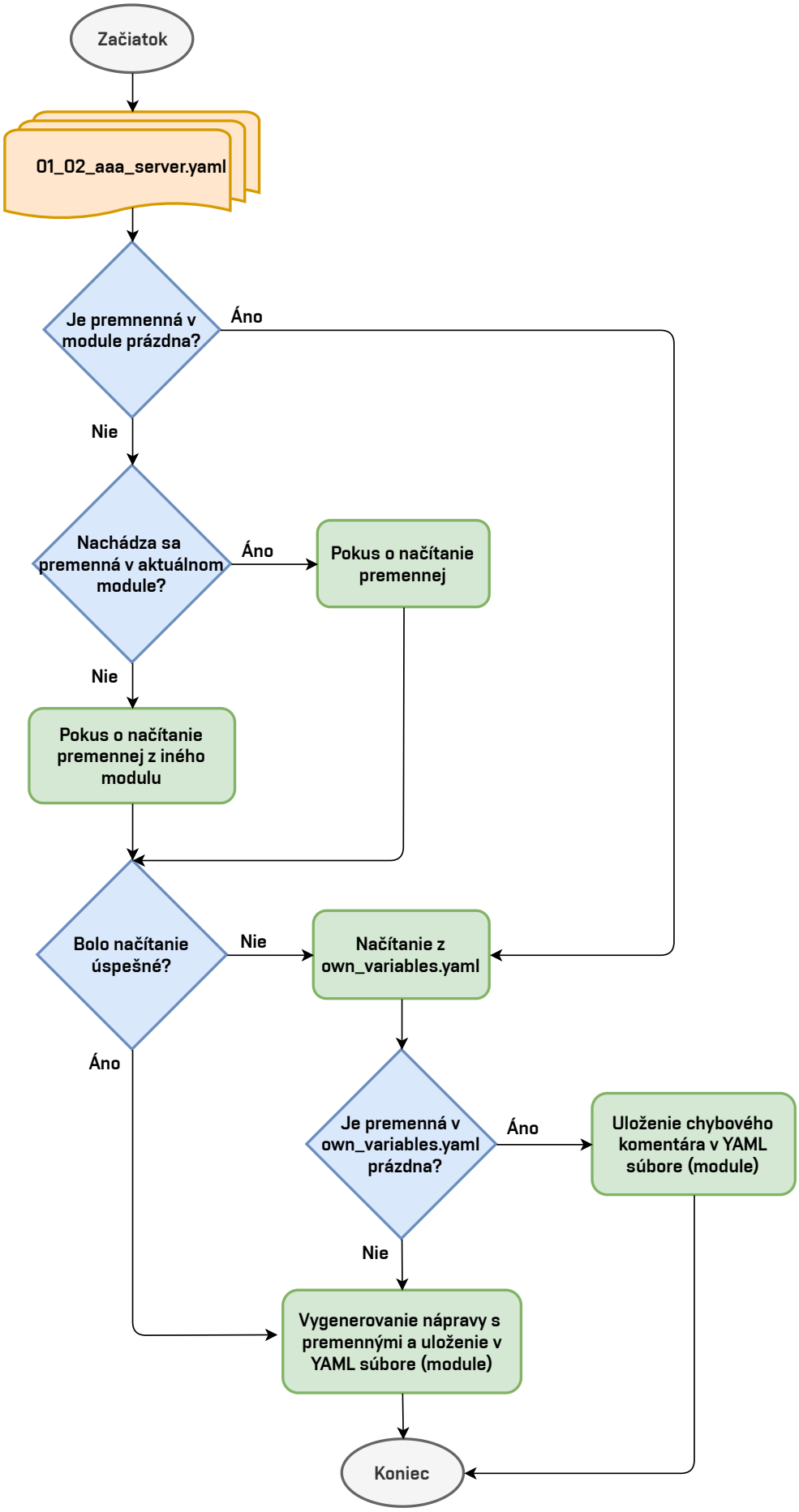
Môže nastať päť situácií s hodnotou premenných v `public_vars`. Pokiaľ je reťazec prázdny, tak sa zoberie hodnota z `own_variables.yaml`. Pokiaľ prázdny nie je, tak pri úspešnom nájdení pomocou regulárneho výrazu je možné vytiahnuť z tejto zhody určité skupiny, príkladom je premenná `aaa_type: "group(1)"` vo výpise 5.2. Pokiaľ pred `group(1)` nie je bližšie špecifikované, ku ktorému regulárnemu výrazu z `regex_cmd` patrí, tak táto skupina musí byť rovnaká vo všetkých regulárnych výrazoch, ktoré sú definované v `regex_cmd`. To je aj dosiahnuté ak sa pozrieme na všetky tri regulárne výrazy v tomto výpise. Následne môže nastať prípad, kedy potrebujeme niektorú premennú vyplniť iba v prípade úspešného nájdenia jedného z regulárnych výrazov. Vtedy použijeme konštrukciu `regex_cmd[0].group(2)`, ktorá hovorí, že skupina 2 z regulárneho výrazu bude do premennej daná iba v prípade, že zhodu našiel prvý regulárny výraz (indexovanie od nuly). Predposledná možnosť je sa odkázať na premennú z iného modulu, kde bola premenná vopred vyplnená alebo úspešne doplnená, napríklad `01_03_aaa_server_group.yaml.aaa_server_type`. Túto mož-

nosť je dobré využiť pokiaľ informácia z jedného príkazu je nutná na vytvorenie nadväzujúceho doplňujúceho príkazu, ktorý je v inom module. Posledná možnosť je natvrdo vyplniť hodnotu nejakým reťazcom.

Ako bolo už spomenuté, súbor `own_variables.yaml` obsahuje informácie pre celý jeden workspace, môže sa stať, že pre nejaké zariadenie môžeme chcieť špecifické nastavenie odlišné od definovaného v tomto súbore. Toto je možné docieliť editáciou samotného modulu v danom adresári zariadenia, ktorý vyhľadáva a napravuje nedostatok, ale je tak možné učiniť až po pustení programu s argumentom `analyze`.

Výpis 5.4: Ukážka skráteného príkladu vyplneného konfiguračného súboru `own_variables.yaml` definujúceho premenné potrebné na generovanie nápravy

```
---
aaa_type: "radius"
aaa_server_name: "CORP_RADIUS"
...
ssh_port: "2223"
...
syslog_ipv4: "10.0.0.140"
...
ospf-router-id: "1.1.1.1"
...
allowed_vlan_on_trunk: "10,20,30"
...
...
```



Obr. 5.1: Vývojový diagram opisujúci hľadanie premenných pre nápravu

5.3 Popis fungovania a diagramy funkcií

Program je vytvorený s orientáciou na budúcu možnú rozšíriteľnosť, prenositeľnosť, modularitu a udržovateľnosť kódu. Z tohto dôvodu sa nejedná o jeden skript, ktorý ma v sebe natvrdo napísané regulárne výrazy na hľadanie nedostatkov, reakciu na ne a rôzne ďalšie minoritné výstupy, ale obsahuje "moduly". Tieto moduly, viď 5.2.2 sú konfiguračnými súbormi využívajúcimi syntax a sémantiku jazyka YAML a definujú správanie programu pri nájdení, odstraňovaní a notifikovaní o nedostatku na analyzovanom zariadení. Samozrejme tento koncept generuje určitú vyššiu réžiu na ukladanie nálezov, ale značne tým zvyšuje modularitu, znovupoužiteľnosť a budúcu rozšíriteľnosť.

V prechádzajúcich kapitolách a v adresárovej štruktúre opísanej v kapitole 4.4 bolo často spomenuté slovo "workspace". Jedná sa v podstate o zložku s vyexportovanými konfiguráciami, ktoré patria pod jednu sieť. Záleží od používateľa programu, ktoré všetky zariadenia považuje za jednu sieť a chce ich zanalyzovať naraz. Zároveň je však odporúčané dávať do jednej workspace zložky ideálne zariadenia operujúce na rovnakej vrstve hierarchického modelu, t.j. napríklad všetky zariadenia z danej testovanej siete, ktoré sa používajú na prístupovej (access) vrstve. Program síce obsahuje algoritmus popísaný v 5.3.2 na rozpoznanie zariadenia a zatriedenie do vrstvy hierarchického modelu, no jeho spoľahlivosť nie je 100%-ná. Preto existuje prepínač, ktorý definuje, že všetky zariadenia v danej zložke (workspace) patria iba do jednej určitej vrstvy. A z tohto dôvodu je odporúčané dávať do analyzovanej zložky ideálne zariadenia operujúce na rovnakej vrstve. Je nutné taktiež podotknúť, že nie je nutné ako názov zložky používať slovo "workspace", užívateľ si tento názov môže zvoliť ľubovoľne, ide iba o demonštráciu. Pri analyzovaní siete organizácie VUT FEKT sa môže zvoliť názov zložky "sieť_VUT_FEKT".

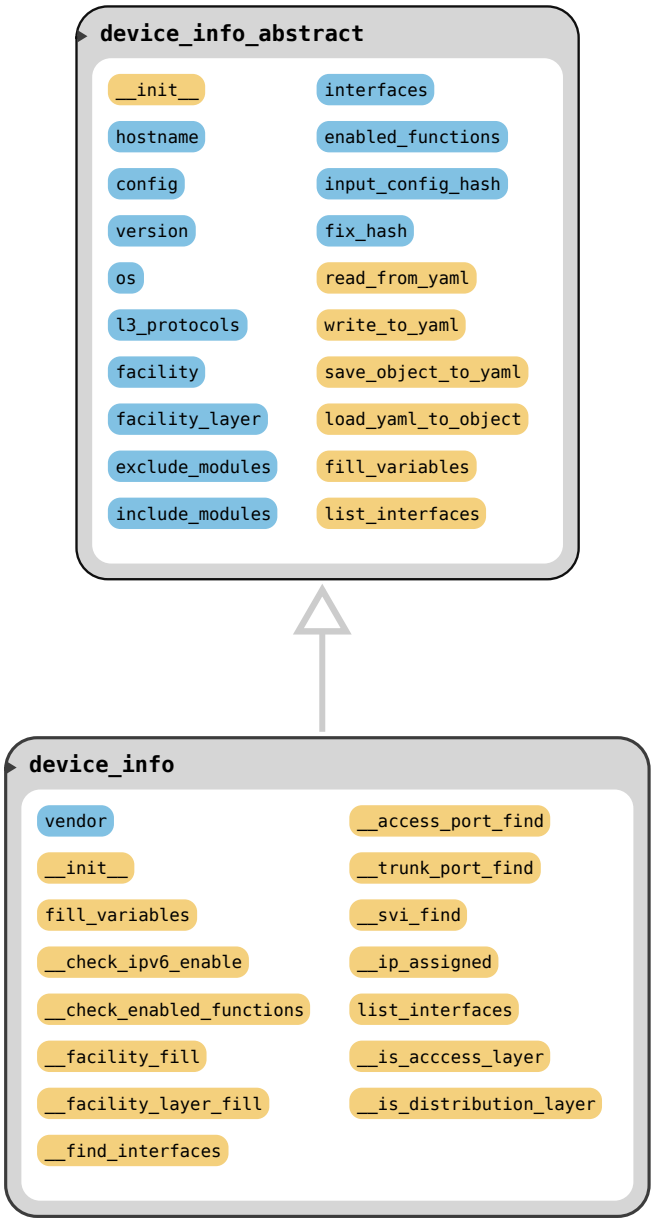
5.3.1 Vytvorené triedy

Program pre svoju prácu využíva tri dôležité triedy. Prvou je abstraktná trieda `device_info_abstract`, z ktorej budú dediť všetky ďalšie triedy `device_info`. Triedy `device_info` respektíve súbor `device_info.py` bude musieť byť vytvorený pre každého výrobcu zvlášť, pričom stačí využiť dedičnosť, podediť premenné z `device_info_abstract`, špecifikovať triednu premennú `vendor` a povinne implementovať metódy `fill_variables`, `list_interfaces`.

Druhá trieda je `device_info` 5.2 a ako už názov napovedá, tak je názov rovnaký ako YAML súbor `device_info.yaml`. Táto trieda obsahuje metódy, v obrázku označené žltou farbou, ktoré zisťujú základné informácie o zariadení a následne ich ukladajú do inštančných premenných ilustrovaných modrou farbou v triede

`device_info_abstract`. Niektoré metódy a všetky inštančné premenná sú zdedené z abstraktnej triedy `device_info_abstract`. Metódy začínajúce podtržníkmi sú pseudo privátne, metóda `__init__` je konštruktor a ostatné metódy sa môžu volať aj mimo tela samotnej triedy. Špeciálnym prípadom je premenná `vendor`, ktorá je triedna. Treba ju preto definovať v každom súbore `device_info.py` u jednotlivých výrobcov zariadení. Napísanie povinných dvoch metód zmienených vyššie, ktoré sú zdedené z `device_info_abstract`, je nutné z dôvodu rozdielnej syntaxe jednotlivých výrobcov a operačných systémov bežiacich na zariadeniach. Je to v podstate jediné potrebné programovanie, ktoré je nutné spraviť pri rozšírení podpory programu na ďalších výrobcov. Ostatné prídavné moduly sa definujú iba pomocou konfiguračných súborov YAML, ktorého príklad je uvedený vo výpise 5.2.2.

Ukladanie tohto objektu do YAML súboru je nutné z dôvodu, že samotný program sa spúšťa v štyroch fázach a je žiadúce raz získané údaje uložiť, ako ich znovu vyhľadávať vo vyexportovaných nastaveniach zariadení. V prvotnej fáze spustenia programu s argumentom `analyze` sa získané informácie uložia do YAML konfiguračného súboru `device_info.yaml` za pomoci metód `save_object_to_yaml` a následne `write_to_yaml`. Analogicky pri spustení programu so zvyšnými troma argumentami sa na prácu s daným zariadením využijú metódy na načítanie `read_from_yaml` a `load_yaml_to_object`.



Obr. 5.2: Objekt `device_info`. Inštančné/triedne premenné sú označené modrou farbou a metódy žltou farbou.

Treťou triedou je `yaml_module`, ktorá slúži na uchovanie informácií, ale aj ako zdroj informácií pri hľadaní nedostatkov a obsahuje dáta potrebné na zjednanie nápravy. Obsahuje iba štyri metódy, ktoré sú nutné pre prácu s YAML súbormi a boli popísané v predchádzajúcom odstavci. Je vidieť, že počet inštančných premenných je značne veľký, to je dané jednak generalizáciou príkazov, ktoré sieťové zariadenie konfigurujú, ale aj orientáciou a rozšírením programu do budúcnosti, kde sa ráta vyššia miera interakcie od administrátora, hlavne označovanie falošne pozitívnych

nálezov, ich komentovanie a mnohé ďalšie funkcie. Tieto funkcie sú dostupné aj v súčasnosti, no komentáre sú editovateľné v YAML súbore iba pomocou textového editora. V budúcnosti sa počíta aj s vytvorením backend-u v jazyku Python a editácia vo webovom prehliadači.

Obr. 5.3: Objekt `yaml_module`. Inštančné/triedne premenné sú označené modrou farbou a metódy žltou farbou.

5.3.2 Automatické zistenie vrstvy

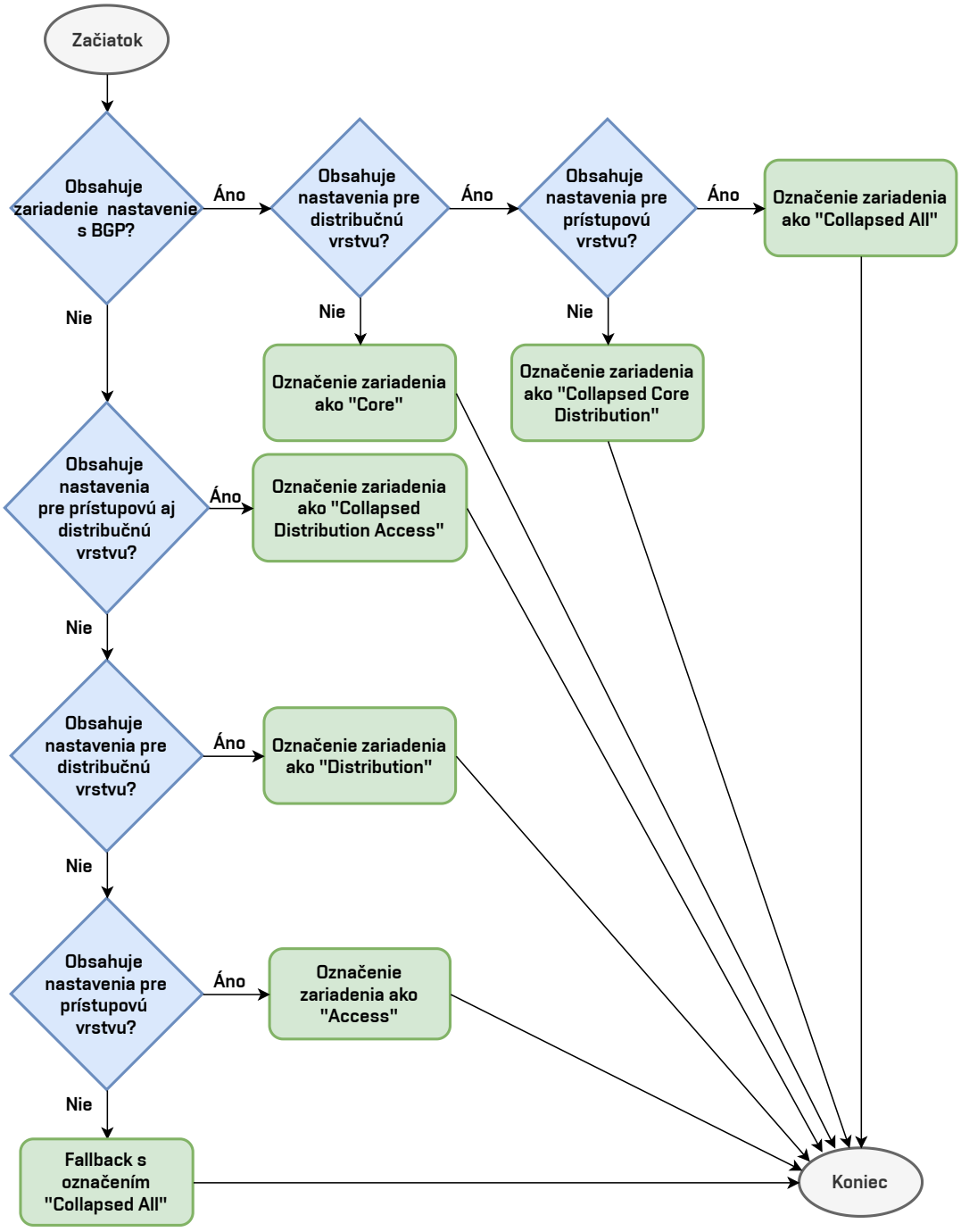
Ako už bolo spomenuté viackrát, tak pre minimalizáciu falošne pozitívnych správ treba spúšťať iba tie moduly, ktoré sú nutné pre dané zariadenie na konkrétnej vrstve hierarchického modelu. Z tohto dôvodu pri spustení programu s parametrom `analyze` príde k automatickému zisteniu vrstvy hierarchického modelu, na ktorom zariadenie operuje. Algoritmus popisujúci túto činnosť 5.4 nie je však dokonalý. Ako si môžeme všimnúť, tak pre označenie zariadenia ako "Core" predpokladá prítomnosť nastaveného protokolu BGP. Nie však na všetkých "Core" prvkoch treba mať BGP zapnuté, hlavne nie pri malých hraničných sieťach, kde sa zväčša nastavuje východzia brána. Hraničné "Core" zariadenia väčšinou nedisponujú mnohými nastaveniami a je veľmi obtiažne zistiť či ide naozaj o "Core" zariadenie. Niektoré siete zasa využívajú BGP aj ako interný smerovací protokol. Preto nie je v určitých špeciálnych prípadoch dobré sa viazať na tento vytvorený algoritmus, vo všeobecnosti však veľmi dobre funguje na rozoznanie ostatných vrstiev popísaných v kapitole 4.3.

Pri každom spustení programu s argumentom `analyze` je na konzolu zobrazená dvojica hostname zariadenia a automaticky zistená vrstva zariadenia. Teda administrátor má informáciu o tom, do akej vrstvy bolo zariadenie zaradené a prikladá informáciou o tom, kde sa dá ručne zmeniť toto rozhodnutie.

Samozrejme je možné definovať ručne vrstvu editovaním YAML modulu pre želané zariadenia v `device_info.yaml`. Toto však nie je úplne komfortné, a preto je možné pri prvotnom spúšťaní programu s argumentom `analyze` použiť prepínač `-facility_layer` a definovať to pre všetky zariadenia v danom workspace. Z predchádzajúceho teda vyplýva, že je namieste separovať exportované konfigurácie zariadení podľa vrstvy do separátnych workspace a teda nepoužívať jeden workspace pre celú topológiu.

Predpoklady pre rozhodnutie príslušnosti zariadenia k danej vrstve:

- Core – zapnutý protokol BGP
- Distribution – použité dynamické IGP protokoly/použitie protokolov FHRP/ smerovatelné porty/ip adresa na portoch
- Access – spanning tree edge port/spanning tree BPDU guard/špecifikovaná access VLAN/802.1x/port security(maximum MAC adries na port)



Obr. 5.4: Vývojový diagram opisujúci automatické zistenie vrstvy

5.3.3 Pridávanie modulov a nových výrobcov

V prípade rozšírenia a pridania modulu na hľadanie nedostatku pre zariadenia Cisco s IOS je potrebné:

1. Definovanie regulárneho výrazu hľadajúceho nastavenie.
2. Zatriedenie typu hľadaného príkazu podľa 4.2.
3. Vytvorenie YAML modulu, ktorého šablóna je v zložke `examples`.

4. Vytvorenie nutných premenných pre príkazy v súbore `own_variables.yaml`.
5. Pridanie YAML modulu do súboru `modules_by_facility_layer.yaml` k príslušnej vrstve.

V prípade rozšírenia podpory programu pre ďalšieho výrobcu je potrebné:

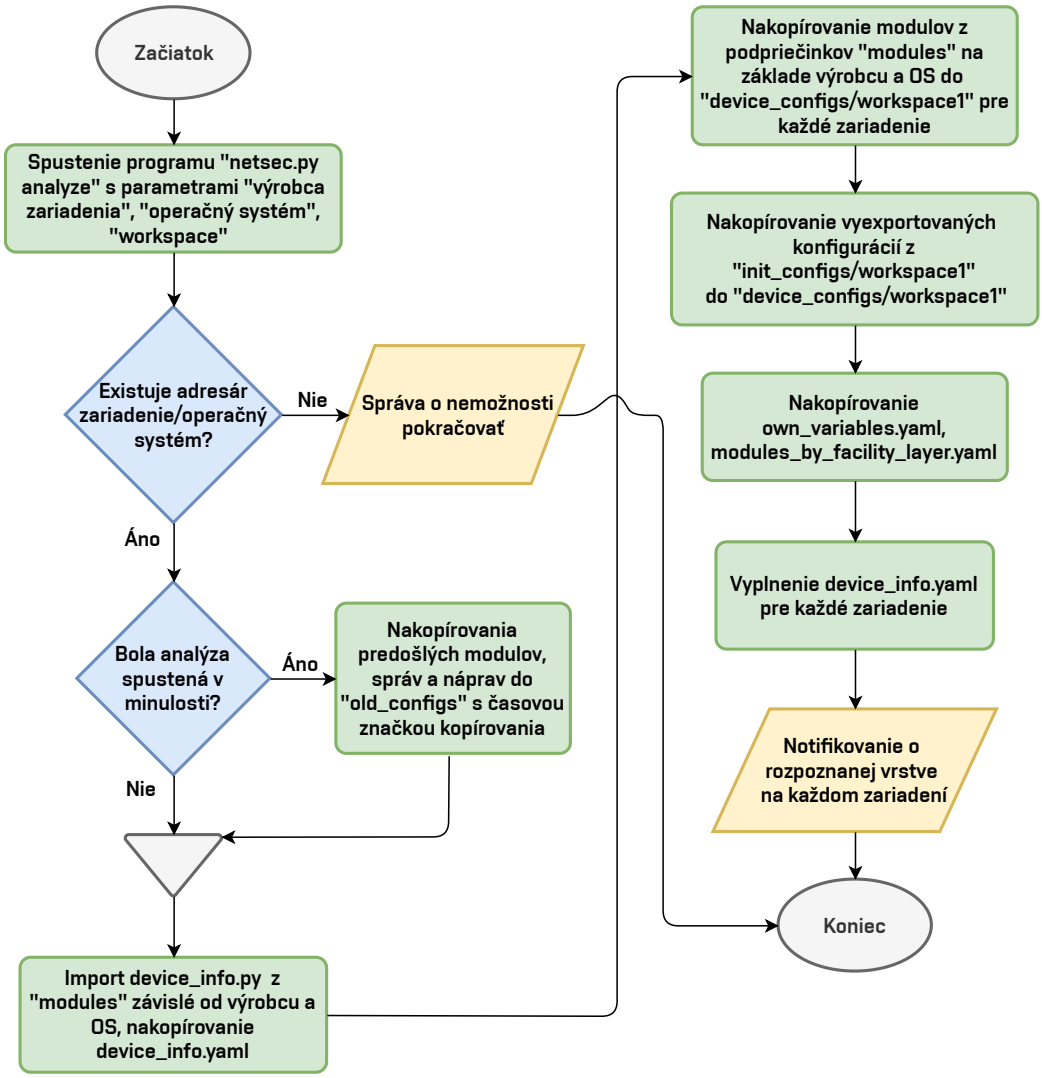
1. Vytvorenie súborovej štruktúry `modules/výrobca/os`.
2. Vytvorenie modulov podľa predchádzajúceho postupu. Moduly by mali odpovedať odporúčaniu z návrhu 4.5.
3. Vytvorenie súboru `device_info.py` dedením `device_info_abstract` a implementovanie nutných metód. Taktiež je potreba vytvoriť metódy na zistenie základných nastavením ktoré sú rozdielne u každého výrobcu a operačného systému, keďže majú inú syntax. Údaje, ktoré je treba zistiť, sú jednotlivé inštančné premenné súboru `device_info_abstract.yaml`.

5.4 Spúšťanie programu

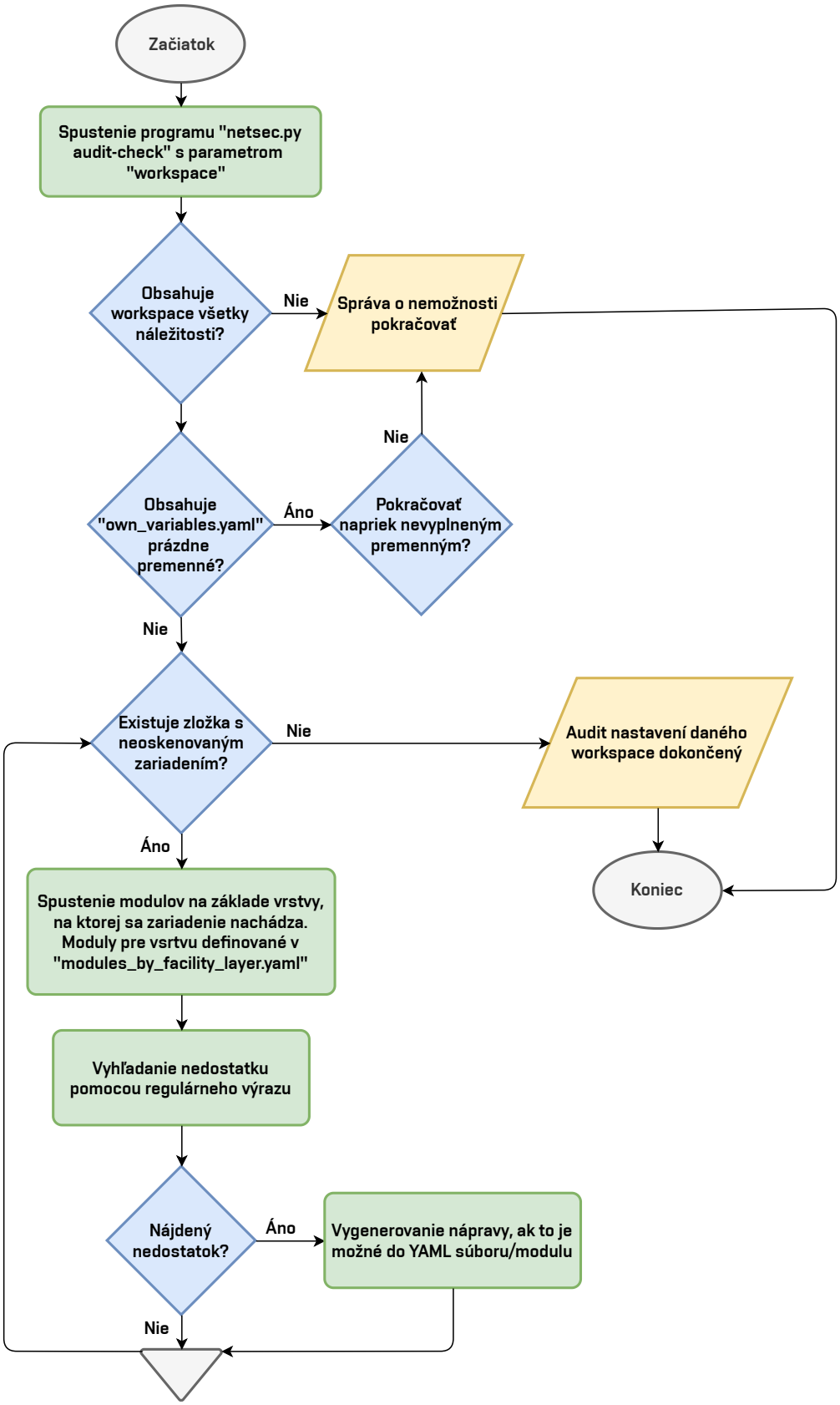
Beh programu sa rozdeľuje do štyroch fáz. Je to z dôvodu, že pri naplnení niektorých súborov sa môže alebo je potrebné editovať ich obsah.

1. Inicializácia – je dôležitá pre zistenie základných informácií o zariadeniach v definovanom workspace. Zistené údaje sú napríklad jeho meno, role jednotlivých rozhraní, zatriedenie zariadenia do hierarchického modelu siete, a tým zistenie, ktoré moduly sa budú spúšťať v nasledujúcej fáze. Zjednodušený postup práce programu tejto časti je znázornený pomocou vývojového diagramu 5.5.
2. Hľadanie nedostatkov a generovanie nápravy – pre každé zariadenie je spustený zoznam modulov zodpovedajúcich vrstve, na ktorej zariadenie pracuje. Je dobré po úvodnej analýze vyplniť súbor `own_variables.yaml`, kde sa nachádzajú premenné potrebné pre nápravu v prípade príkazov, ktoré potrebujú dodatočné informácie. Diagram 5.6 opisujúci hlavnú časť programu je veľmi zjednodušený, zovšeobecnený a abstraktný. V prípade záujmu je možné nahliadnuť do zdrojového súboru `netsec.py` do metód `audit_check` a `audit_analyze_module`.
3. Generovanie záverečnej správy – diagram pre túto časť programu nie je potrebný nakoľko ide o priamočiaru prácu. Otvoria sa všetky moduly v zložke každého zariadenia, zistí sa stav hľadania nedostatkov a všetky údaje sa použijú na vytvorenie záverečnej HTML správy aj so štatistickými informáciami. Príklad tejto správy je k nahliadnutiu v kapitole 5.5.
4. Vyexportovanie súborov pre nápravu – pre túto časť programu nie je taktiež nutná vizualizácia, keďže sa zoberú vygenerované reťazce nápravy z každého YAML modulu v adresári pre konkrétne zariadenie a pridajú sa do textového

súboru, ktorý sa následne bude aplikovať na zariadenie. Tento nápravný textový súbor je vygenerovaný pre každé zariadenie. Môže byť aplikovaný ručne pomocou nakopírovania do konzole alebo v prípade existencie automatických nasadzovacích nástrojov ako Ansible, sa táto činnosť môže spraviť automatizovane.



Obr. 5.5: Inicializácia programu pomocou argumentu `analyze`.



Obr. 5.6: Veľmi zjednodušený pohľad na prácu hlavnej časti programu zodpovednú za nájdenie nedostatkov a vygenerovanie ich nápravy spustením pomocou argumentu `audit-check`.

5.5 Správa s nedostatkami

Po úspešnej analýze nastavení zariadení je možné pomocou argumentu `generate_report` vytvoriť správu o analýze s nájdenými nedostatkami. Správa obsahuje základné informácie o zariadení zo súboru `device_info.yaml`, ktorý ako už bolo spomenuté, slúži na uchovanie premenných z triedy `device_info`. Naviac je správa doplnená o čas jej vygenerovania. Správa je zámerne v anglickom jazyku z dôvodu univerzálnosti a tiež kvôli predpokladanému budúcemu využitiu programu ďalšími osobami.

Správa obsahuje aj štatistiku úspešnosti. Zaujímavou časťou je koeficient váženého skóre (Weighted score coefficient). Nasledujúca tabuľka reprezentuje priradenie váhy úspešným a neúspešným nálezom:

Úspešnosť hľadania nastavenia	Závažnosť	Váha	Premenná udávajúca početnosť výskytov
Úspech	-	1	počet_úspech
Neúspech	Notice	1	počet_notice
Neúspech	Low	0.75	počet_low
Neúspech	Medium	0.5	počet_medium
Neúspech	High	0.25	počet_high
Neúspech	Critical	0	počet_critical

Tab. 5.1: Váhy nálezov

Výpočet koeficientu je nasledovný:

$$normalizovaný\_diel = 100/počet\_všetkých\_testovaných\_modulov$$

$$suma\_koeficientov = 0 * počet\_critical + 0.25 * počet\_high + 0.5 * počet\_medium +$$

$$+ 0.75 * počet\_low + 1 * počet\_notice + 1 * počet\_úspech$$

$$koeficient\_váženého\_skóre = suma\_koeficientov * normalizovaný\_diel$$

V správe o analýze nasledujú jednotlivé oblasti, ako Authentication, Authorization, Accounting; SSH a podobne. Každá oblasť obsahuje názov spusteného modulu, závažnosť (Severity) nastavenia alebo nedostatku, informáciu o preskočení testovania (Skipped), informáciu o nemožnosti nájsť alebo vykonať nápravu (Cannot determine search/fix) a výsledný stav testu (Status). Testovanie môže byť preskočené (Skipped) pokiaľ sa na zariadení nevyskytuje nejaké nastavenie, alebo L3 protokol, čo súvisí s premennými `l3_protocols` a `enabled_functions`, ktoré boli opísané v kapitole 5.2.1. Stĺpec `Cannot determine search/fix` nadobúda hodnotu `True` pokiaľ

nevie program zjednať nápravu pre chýbajúce premenné do príkazu, prípadne pri nenájdení nutného kontextu alebo rozhrania, ktoré musí mať určité nastavenie, napríklad byť prístupovým (access) portom. Následne pri každej položke sa zobrazujú ďalšie dodatočné informácie. Pri nedostatku sa zobrazia rozhrania, ktoré nastavenie absentujú, kontexty, v ktorých bola nájdená chyba a tiež sekvencia príkazov na zjednanie nápravy. Ďalšími údajmi môžu byť nastavenia, ktoré boli úspešne nájdené a rôzne komentáre k zisteniam, ako aj upozornenia a usmernenia. Ilustračná skrátená záverečná správa je dostupná na obrázku nižšie.

Obr. 5.7: Príklad ilustračnej skrátenej záverečnej správy s nedostatkami

AUDIT REPORT

Hostname:	SW1_DISTACC
Config file name:	5
Config hash:	10b09a99888cd890f6d8bb5b9b220c53092b0579
Vendor:	cisco
OS:	ios
Facility layer:	collapsed_distribution_access
L3 protocols:	ipv4,ipv6
Enabled functions:	rip,eigrp
Date:	19/04/2020_14:52

SUCCESS STATISTICS

Weighted score coefficient:	65 out of 100
Successful checks:	49.0%
Unsuccessful checks:	51.0%
Total number of relevant started checks:	157
Successful checks:	76
Unsuccessful 'Critical':	22
Unsuccessful 'High':	29
Unsuccessful 'Medium':	25
Unsuccessful 'Low':	4
Unsuccessful 'Notice':	1

Authentication, Authorization, Accounting

No.	Name	Severity	Skipped	Cannot determine search/fix	Status
1	AAA new model/ Enabled AAA	High	False	False	Successful ✔
	Right configuration setting found in: aaa new-model				
2	AAA server set	High	False	False	Successful ✔
	Right configuration setting found in: radius server MY_SERVER address ipv4 192.168.1.1 auth-port 1645 acct-port 1646 timeout 1 non-standard key AAA				
3	AAA server group	High	False	False	Successful ✔
	Right configuration setting found in: aaa group server radius MYGROUP server name MY_SERVER				

SSH

No.	Name	Severity	Skipped	Cannot determine search/fix	Status
1	SSH enable on management connection	Critical	False	False	Error ✘

Right configuration setting found in:
line vty 0 4
transport input ssh
line vty 6
transport input ssh

Found error in context(s):
line vty 5
line vty 10 11

Generated fix:
line vty 5
transport input ssh

line vty 10 11
transport input ssh

VLAN

1	Native vlan different than 1	Critical	False	False	Error ✘

Right configuration setting found in:
interface Ethernet2/0
switchport trunk native vlan 100

Found port(s) with error:
Ethernet2/1
Ethernet3/2

Generated fix:
interface Ethernet2/1
switchport trunk native vlan 100
interface Ethernet3/2
switchport trunk native vlan 100

Network Mapping

No.	Name	Severity	Skipped	Cannot determine search/fix	Status
1	CDP globally disabled	Critical	False	False	Successful ✔

Right configuration setting found in:
no cdp run

2	CDP disabled on interface(s)	Critical	False	False	Matched by equivalent ✔

Comment:
Module was marked as matched by equivalent by module 11_01_cdp_run_disabled.yaml

3	LLDP globally disabled	Critical	False	False	Successful ✔
4	LLDP receive disabled on interface(s)	Critical	False	False	Not relevant ✔

Comment:
Module 11_04_lldp_receive.yaml configured to run after 11_03_lldp_run_disabled.yaml with status error but that module has not run yet with specified cmd_match_status, 11_04_lldp_receive.yaml will not run

5.6 Požiadavky programu

Na bezproblémovú prácu s programom sú potrebné tieto náležitosti:

1. Python 3.7.3 alebo vyšší
2. Python modul `ruamel.yaml`
3. Python modul `pdfkit` (voliteľný) – pri nenainštalovaní nebudú vygenerované PDF správy, ale iba HTML správy.
4. Program `wkhtmltopdf` (voliteľný) – pri nenainštalovaní nebudú vygenerované PDF správy, ale iba HTML správy.

6 Testovanie

Testovanie programu, a to predovšetkým funkčnosti jednotlivých YAML modulov prebiehalo postupne pri ich zostavovaní. Funkčnosť jednotlivých metód a funkcií bola testovaná rôznymi priaznivými, ako aj chybovými vstupmi, ktoré by mohli nastať zlým nastavením programu, ale aj nekonzistentným vstupom. Pre testovanie YAML modulov bolo nevyhnutné získať čo najviac kombinácií nastavení, na čo bol použitý program GNS3, ktorý dokáže simulovať sieťové zariadenia a vyexportované nastavenia týchto zariadení môžu byť následne testované. Ako už bolo spomenuté v prechádzajúcich kapitolách, tak na vyhľadávanie nedostatkov boli použité regulárne výrazy. Dĺžka týchto regulárnych výrazov v tomto programe je od pár desiatkov znakov až po cca. 1500 pri komplikovanom filtračnom pravidle. Niekedy na prvý pohľad komplikované regulárne výrazy sú nutné, pretože zápis niektorých príkazov má množstvo variácií či už v poradí jednotlivých častí, alebo rozdielnej syntaxe naprieč verziami operačného systému. Validácia a ladenie regulárnych výrazov prebiehala za pomoci stránky `www.regex101.com`. Na validáciu bolo nutné mnohokrát rôznym spôsobom nakombinovať časti príkazu a taktiež vytvoriť zdanlivo fungujúci príkaz, v ktorom sa nemohla nájsť zhoda.

6.1 Testovacie prostredie

CPU:	Intel Core i5 6200U
RAM:	12 GB
OS:	Debian 10 (Buster)
Python:	3.7.3
Knižnice:	ruamel.yaml 0.16.10, pdfkit 0.6.1
SW:	GNS3 2.1.21, wkhtmltopdf 0.12.5
Sieťový HW:	Cisco IOU L2 15.2d, Cisco IOU L3 15.5(2)T, Cisco IOSvL2 15.2.1

Tab. 6.1: Testovacie prostredie

6.2 Príklad spustenia so vzorovými nastaveniami

Pre otestovanie a demonštráciu funkčnosti výsledného programu bola v nástroji GNS3 vytvorená topológia a následne vyexportované nastavenia jednotlivých sieťových prvkov. Tieto nastavenia dostupné v `init_configs/topologia_diplomova_praca`

budú použité v nasledujúcom návode na demonštračné spustenie. Vyexportované nastavenia obsahujú zámerne chyby, aby bolo možné otestovať jednak vyhľadávanie nedostatkov v nastaveniach, ale aj úspešnosť aplikovania generovaných náprav. Úspešnosť aplikovaných náprav bude verifikovaná opätovným spustením programu, no s vyexportovanými konfiguráciami po aplikácií nápravných príkazov. Pri spúšťaní je treba byť v zložke s hlavným skriptom `netsec.py` a spúšťať ho z tohto umiestnenia. V prípade použitia nastavení z testovacej konfigurácie stačí použiť nasledujúci príkaz nižšie. Pri analýze iných nastavení je nutné vytvoriť novú zložku v `init_configs` napríklad `topológia_FEKT_VUT` a do nej umiestniť vyexportované konfigurácie. Viac o tejto problematike v 5.3. Keďže kvôli časovej náročnosti je doposiaľ prítomná podpora iba pre zariadenia Cisco s operačným systémom IOS, tak je jedinou možnosťou špecifikovať prepínače s nasledujúcimi argumentami `--vendor "cisco" --os "ios"`. Pri opätovnom spustení prvotnej analýzy, teda zistení základných informácií o zariadeniach, nie samotnom vyhľadávaní nedostatkov, je občas žiadúce ponechať v minulosti vyplnené premenné v `own_variables.yaml` z predchádzajúceho spustenia programu. Toto sa vykonáva pomocou prepínaču `--keep-own-vars`. Každá časť spustenia obsahuje ukazovateľa postupu v percentách, aby bolo zrejmé, že program niečo vykonáva, podrobnejšie výpisy nie sú prítomné, pretože by samostatné nadmerné vypisovanie mohlo spomaľovať beh programu. Po poslednom sprocesovanom zariadení sa za ukazovateľom ukáže namiesto percent hlásenie `Done!`. Pri opätovnom spustení programu s argumentom `analyze` sa predchádzajúce zistenia, správy, moduly s informáciami automaticky presunú do umiestnenia `old_configs/topologia_diplomova_pr` kde sa vytvorí vždy zložka s aktuálnym časom.

Výpis 6.1: Spustenie úvodnej analýzy topológie

```
./netsec.py analyze --workspace "topologia_diplomova_praca"
  --vendor "cisco" --os "ios"

[========] Done!

Hostname                        Facility layer(Automatically set)
Access_SW1                      access
Access_SW2                      access
Access_SW3                      access
BRANCH_Office                   collapsed_all
Core                            core
Dist_Router                     distribution
Dist_SW1                        distribution
Dist_SW2                        collapsed_distribution_access

To edit facility layer go to diplomova_praca/src/device_configs/
topologia_diplomova_praca to corresponding folder named
by hostname and change variable 'faciliy_layer' in
device_info.yaml

Workspace successfully analyzed!
```

V ďalšej fáze sa spúšťa vyhľadávanie a generovanie náprav nájdených nedostatkov, pokiaľ je to možné. Ešte pred tým je však nutné v aktuálnom workspace, teda v `device_configs/topologia_diplomova_praca` vyplniť východzie nastavenia premenných v súbore `own_variables.yaml`. Pre urýchlenie je takýto súbor predvyplnený v zložke `testing` a stačí ho nakopírovať a prepísať v adresári `device_configs/topologia_diplomova_praca`. V prípade nejakej nevyplnenej premennej v tomto súbore je administrátor notifikovaný a musí potvrdiť, že nevyplnením premenných nemusí generovanie nápravy fungovať ako má. Z dôvodu anonymizácie je možné neukladať úspešné nájdenia a v následne vygenerovanej správe informovať iba o prítomnosti a neprítomnosti nastavenia pomocou prepínača `--hide-match`.

Výpis 6.2: Spustenie vyhľadávania nedostatkov a generovania nápravy

```
./netsec.py audit-check --workspace "topologia_diplomova_praca"

  [========] Done!

  Audit check was done on workspace!
```

Následne sa môže spustiť vygenerovanie záverečných správ pre každé zariadenie. Z dôvodu anonymizácie je tiež možné nezobrazovať úspešné nájdenia v správe a informovať iba o prítomnosti a neprítomnosti nastavenia pomocou prepínača `--hide-match`. Použitie tohto prepínača nepodmieňuje jeho použitie v predchádzajúcej časti, ak v predchádzajúcej časti nebol použitý tento prepínač, je možné nevypisovať tieto informácie, napriek ich zapísaniu do YAML modulov. Naopak ak bol v predchádzajúcej časti použitý, tak zhody nájdené v konfigurácií sú vymazané a nepôjdu v žiadnom prípade zobraziť a správa bude obsahovať iba informáciu o prítomnosti alebo nedostatku nastavenia, nie však jeho skutočnú hodnotu v konfigurácií. Pokiaľ je na počítači, kde prebieha analýza nainštalovaný Python modul `pdfkit` a program `wkhtmltopdf`, tak sú v adresári `device_configs/topologia_diplomova_praca/reports` dostupné okrem správ vo formáte HTML, aj správy v PDF. V prípade, že tieto nepovinné závislosti nie sú prítomné, je možné HTML správu zobraziť vo webovom prehliadači a PDF súbor vytvoriť pomocou vytlačenia "Vytlačiť ako PDF".

Výpis 6.3: Spustenie vygenerovania záverečných správ

```
./netsec.py generate-report --workspace "
   topologia_diplomova_praca"

[========] Done!

Reports were generated on workspace!
```

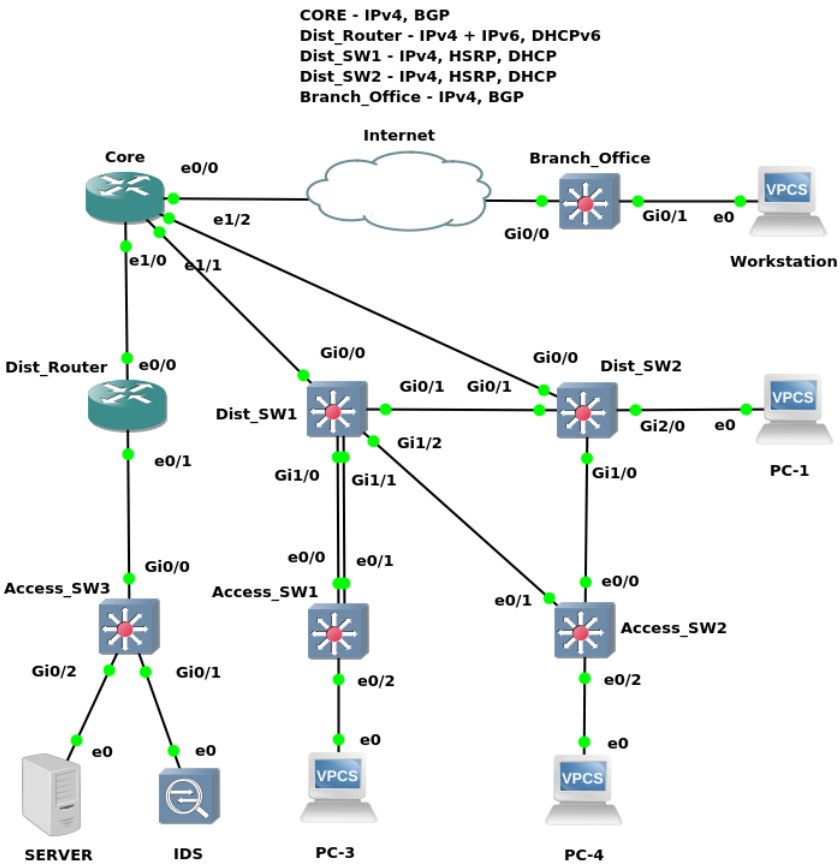
Posledným spustením programu zabezpečíme vytvorenie nápravných textových súborov, ktoré budú aplikovateľné na problémové zariadenia. Pre každé zariadenie sa vygeneruje textový súbor do zložky `device_configs/topologia_diplomova_praca/current_fixes`.

Výpis 6.4: Spustenie vygenerovania nápravných textových súborov

```
./netsec.py generate-fix --workspace "topologia_diplomova_praca"

[========] Done!

Fixes were generated on workspace!
```



Obr. 6.1: Testovacia topológia, pre ktorú sú dostupné vyexportované nastavenia zariadení.

Záver

Cieľom tejto diplomovej práce bol návrh a následná implementácia programu na nájdenie bezpečnostných a prevádzkových nedostatkov v sieťových zariadeniach, ako aj ich náprava pomocou generovania opravnej konfigurácie. Z tohto dôvodu bola naštudovaná problematika bezpečnosti a prevádzky sieťových zariadení a ich správna konfigurácia. Z množstva dostupnej literatúry, štandardov a odporúčaní bol vytvorený zoznam odporúčaní, na základe ktorého boli zostavované YAML moduly pre výsledný program. Tento zoznam odporúčaní bol rozšírený aj o hodnotenie závažnosti nedostatkov a priradenie prvkov zoznamu k relevantným zariadeniam v hierarchickom modely siete. Jedná sa teda o unikátne riešenie medzi bezplatnými zoznamami odporúčaní. Tento zoznam môže byť použitý aj na rozšírenie programu o podporu iných výrobcov, ale aj separátne bez akéhokoľvek využitia v programe. Vzhľadom na časovú náročnosť boli vytvorené moduly zatiaľ pre zariadenia značky Cisco. Bolo nutné zanalyzovať niekoľko stovák príkazov a ich povolených kombinácií, vytvoriť pre ne korešpondujúce regulárne výrazy a následne vytvoriť viac ako 230 YAML modulov zodpovedných za nájdenie problémov v konfiguráciách.

Ako implementačný jazyk bol využitý Python 3.7, ktorý zaisťuje prenositeľnosť programu na viaceré platformy. Program na rozdiel od konkurencie umožňuje rozšírenie vďaka modularite aj na ďalších výrobcov sieťových zariadení a pridáva kontrolu aj pre topológie využívajúce IPv6. Taktiež rešpektuje hierarchický model siete, a teda generuje oveľa menej falošne pozitívnych správ, keďže kontroluje iba nastavenia typické pre danú vrstvu, na ktorej zariadenie operuje. Jeho výstupom je okrem iného aj prehľadná správa o kontrole zobraziteľná v PDF alebo vo webovom prehliadači. Naviac ako jediný z porovnávaných bezplatných riešení umožňuje automatické vygenerovanie nápravy, pokiaľ je to z charakteru príkazu možné.

Program bol otestovaný pomocou vyexportovaných konfigurácií z viacerých testovacích topológií vytvorených v nástroji GNS3. Je však žiadúce aplikáciu otestovať komplexnejšie aj na reálnych a rozsiahlejších topológiách pre čo najlepšie vyladenie jej funkčnosti.

Nástroj je možné vďaka modularite v budúcnosti rozšíriť aj o podporu na ďalších výrobcov ako Juniper a HP. Využiteľná by bola taktiež možnosť editovať záverečné správy pridaním back-endu pre HTML správy, teda označovaním falošne pozitívnych nálezov a komentovaním nálezov priamo vo webovom prehliadači. Tiež by bolo možné program rozšíriť o podporu notifikovania nedostatkov a chýb na základe známych hrozieb pre aktuálne bežiace verzie operačných systémov.

Literatúra

[1] MILKOVICH, Devon. 13 Alarming Cyber Security Facts and Stats. In: *Cybint* [online]. 3.12.2018 [cit. 2019-11-08]. Dostupné z: https://www.cybintsolutions.com/cyber-security-facts-stats/

[2] MCMILLAN, Troy. *CCNA security study guide: exam 210-260*. Indianapolis, Indiana: Sybex, a Wiley Brand, 2018. ISBN 978-111-9409-939.

[3] VYNCKE, Eric a Christopher PAGGEN. *LAN switch security: What hackers know about your switches*. Indianapolis, IN: Cisco Press, 2008. ISBN :978-1-58705-256-9.

[4] STALLINGS, William. *Network security essentials: applications and standards*. 4th ed. Boston: Prentice Hall, 2011. ISBN 978-0-13-610805-4.

[5] JACKSON, Chris. *Network security auditing*. Indianapolis, IN: Cisco Press, 2010. Cisco Press networking technology series. ISBN 978-1-58705-352-8.

[6] Guide for Conducting Risk Assessments: NIST Special Publication 800-30. In: *NIST* [online]. 2012 [cit. 2019-11-08]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf

[7] LAMMLE, Todd. *CCNA: routing and switching : study guide*. Indianapolis, Indiana: SYBEX, [2013]. ISBN 978-1-118-74961-6.

[8] SINGH, Shashank. Cisco Guide to Harden Cisco IOS Devices. In: *Cisco* [online]. 2018 [cit. 2019-11-02]. Dostupné z: https://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html

[9] PEPELNJAK, Ivan. Management, Control and Data Planes in Network Devices and Systems. In: *IpSpace* [online]. 2013 [cit. 2019-11-17]. Dostupné z: https://blog.ipspace.net/2013/08/management-control-and-data-planes-in.html

[10] CIS Cisco IOS 15 Benchmark. In: *Center For Internet Security* [online]. 2015 [cit. 2019-11-02]. Dostupné z: https://www.cisecurity.org/benchmark/cisco/

[11] BARKER, Elaine a Allen ROGINSKY. Transitioning the Use of Cryptographic Algorithms and Key Lengths. In: *NIST* [online]. 2019 [cit. 2019-11-02]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf

[12] Special-Purpose IP Address Registries. In: *IETF* [online]. [cit. 2019-1 2-08]. Dostupné z: https://tools.ietf.org/html/rfc6890

[13] Updates to the Special-Purpose IP Address Registries. In: *IETF* [online]. [cit. 2019-12-08]. Dostupné z: https://tools.ietf.org/html/rfc8190

[14] Special-Use IPv6 Addresses. In: *IETF* [online]. [cit. 2019-12-08]. Dostupné z: https://tools.ietf.org/html/rfc5156

[15] GRÉGR, Matěj a Tomáš PODERMAŃSKI. Bezpečné IPv6: vícehlavý útočník. In: *ROOT.CZ* [online]. 26.2.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-vicehlavy-utocnik/

[16] PODERMAŃSKI, Tomáš a Matěj GRÉGR. Bezpečné IPv6: trable s hlavičkami. In: *ROOT.CZ* [online]. 19.2.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-trable-s-hlavickami/

[17] Implications of Oversized IPv6 Header Chains. In: *IETF* [online]. 2014 [cit. 2019-12-18]. Dostupné z: https://tools.ietf.org/html/rfc7112

[18] KHANDELWAL, Manjul. OSPF Security: Attacks and Defenses. In: *SANOG* [online]. 2016 [cit. 2019-11-04]. Dostupné z: https://www.sanog.org/resources/sanog28/SANOG28-Tutorial_OSPF-Security-Attacks-and-Defences-Manjul.pdf

[19] Understanding BGP TTL Security. In: *PacketLife* [online]. 2009 [cit. 2019-11-30]. Dostupné z: https://packetlife.net/blog/2009/nov/23/understanding-bgp-ttl-security/

[20] GRAESSER, Dana. Cisco Router Hardening Step-by-Step. In: *SANS Institute* [online]. 2001 [cit. 2019-11-02]. Dostupné z: https://www.sans.org/reading-room/whitepapers/firewalls/paper/794

[21] Cisco SAFE Reference Guide. In: *CIsco* [online]. San Jose, CA, 8 Júl 2018 [cit. 2019-11-02]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Security/SAFE_RG/SAFE_

[22] NTP Amplification DDoS Attack. In: *Cloudflare* [online]. [cit. 2019-12-01]. Dostupné z: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/

[23] The NTP FAQ and HOWTO. In: *Network Time Protocol* [online]. [cit. 2019-12-01]. Dostupné z: http://www.ntp.org/ntpfaq/NTP-s-algo-crypt.htm

[24] VLAN Hopping: How to Prevent an Attack. In: *AT&T Cybersecurity* [online]. 2018 [cit. 2019-12-03]. Dostupné z: https://www.alienvault.com/blogs/security-essentials/vlan-hopping-and-mitigation

[25] SATRAPA, Pavel. *IPv6: internetový protokol verze 6*. 4. aktualizované a rozšířené vydání. Praha: CZ.NIC, z.s.p.o., 2019. CZ.NIC. ISBN 978-808-8168-430.

[26] Bezpečné IPv6: příliš mnoho sousedů. In: *ROOT.CZ* [online]. 2015 [cit. 2019-12-08]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-prilis-mnoho-sousedu/

[27] ALSADEH, Ahmad. Augmented SEND: Aligning Security, Privacy, and Usability. In: *RIPE NCC* [online]. 12.5.2015 [cit. 2019-11-02]. Dostupné z: https://ripe70.ripe.net/presentations/67-RIPE70-SEND.pdf

[28] GRÉGR, Matěj a Tomáš PODERMAŃSKI. Bezpečné IPv6 : směrovač se hlásí. In: *ROOT.CZ* [online]. 5.2.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-smerovac-se-hlasi/

[29] PODERMAŃSKI, Tomáš a Matěj GRÉGR. Bezpečné IPv6: zkrocení zlých směrovačů. In: *ROOT.CZ* [online]. 12.2.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-zkroceni-zlych-smerovacu/

[30] Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery. In: *IETF* [online]. [cit. 2019-12-08]. Dostupné z: https://tools.ietf.org/html/rfc6980

[31] PODERMAŃSKI, Tomáš a Matěj GRÉGR. Bezpečné IPv6: když dojde keš. In: *ROOT.CZ* [online]. 12.3.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-kdyz-dojde-kes/

[32] PODERMAŃSKI, Tomáš a Matěj GRÉGR. Bezpečné IPv6: když dojde keš — obrana. In: *ROOT.CZ* [online]. 19.3.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-kdyz-dojde-kes-obrana/

[33] PODERMAŃSKI, Tomáš a Matěj GRÉGR. Bezpečné IPv6: trable s multicastem. In: *ROOT.CZ* [online]. 5.3.2015 [cit. 2019-11-02]. Dostupné z: https://www.root.cz/clanky/bezpecne-ipv6-trable-s-multicastem/

[34] REY, Enno, Antonios ATLASIS a Jayson SALAZAR. MLD Considered Harmful. In: *RIPE NCC* [online]. 2016 [cit. 2019-11-02]. Dostupné z: https://ripe72.ripe.net/presentations/74-ERNW_RIPE72_MLD_Considered_Harmful_v1_light_web.pdf

[35] IPv6 First-Hop Security Configuration Guide. In: *Cisco* [online]. San Jose [cit. 2019-11-02]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_fhsec/configuration/15-1sg/ip6f-15-1sg-book.pdf

[36] BOUŠKA, Petr. Cisco IOS 11 - IEEE 802.1x, autentizace k portu, MS IAS. In: *SAMURAJ-cz* [online]. 2007 [cit. 2019-12-09]. Dostupné z: https://www.samuraj-cz.com/clanek/cisco-ios-11-ieee-802-1x-autentizace-k-portu-ms-ias/

[37] BOUŠKA, Petr. *Cisco IOS 12 - IEEE 802.1x a pokročilejší funkce* In:*SAMURAJ-cz* [online]. 2007 [cit. 2019-11-02]. Dostupné z: https://www.samuraj-cz.com/clanek/cisco-ios-12-ieee-802-1x-a-pokrocilejsi-funkce/

[38] MOLENAAR, René. Cisco IOS features that you should disable or restrict. In: *NetworkLessons.com* [online]. [cit. 2019-11-02]. Dostupné z: https://networklessons.com/uncategorized/cisco-ios-features-that-you-should-disable-or-restrict

[39] BOUŠKA, Petr. Cisco IOS 23 - Autentizace uživatele na switchi vůči Active Directory. In: *SAMURAJ-cz* [online]. 2009 [cit. 2019-11-02]. Dostupné z: https://www.samuraj-cz.com/clanek/cisco-ios-23-autentizace-uzivatele-na-switchi-vuci-active-directory/

[40] VYNCKE, Erik. ND on wireless links and/or with sleeping nodes. In: *IETF* [online]. [cit. 2019-11-02]. Dostupné z: https://www.ietf.org/proceedings/89/slides/slides-89-v6ops-3.pdf

[41] VYNCKE, Erik. IPv6 First Hop Security: the IPv6 version of DHCP snooping and dynamic ARP inspection. In: *Slidde Share* [online]. 2012 [cit. 2019-11-02]. Dostupné z: https://www.slideshare.net/IKTNorge/eric-vyncke-layer2-security-ipv6-norway

[42] GREGR, Matej, Petr MATOUSEK, Miroslav SVEDA a Tomas PODERMANSKI. Practical IPv6 monitoring-challenges and techniques. In: *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, 2011, 2011, s. 650-653. DOI: 10.1109/INM.2011.5990647. ISBN 978-1-4244-9219-0. Dostupné také z: http://ieeexplore.ieee.org/document/5990647/

[43] MARTIN, Tim. IPv6 Sys Admin Style. In: *SlideShare* [online]. 2016 [cit. 2019-11-02]. Dostupné z: https://www.slideshare.net/tjmartin2020/ipv6-sysadmins-63071235

[44] SAFE Overview Guide: Threats, Capabilities, and the Security Reference Architecture. In: *Cisco* [online]. Január 2018 [cit. 2019-11-02]. Dostupné z: https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise/design-zone-security/safe-overview-guide.pdf

[45] AKIN, Thomas. *Hardening Cisco routers*. Sebastopol: O'Reilly, 2002. ISBN 05-960-0166-5.

[46] HUCABY, Dave, Steve MCQUERRY, Andrew WHITAKER a Dave HUCABY. *Cisco router configuration handbook*. 2nd ed. Indianapolis, IN: Cisco Press, 2010. ISBN 978-1-58714-116-4.

[47] Port Knocking. In: *Mikrotik* [online]. [cit. 2019-12-09]. Dostupné z: https://wiki.mikrotik.com/wiki/Port_Knocking

[48] DOOLEY, Kevin a Ian J. BROWN. *Cisco IOS cookbook*. 2nd ed. (Revised and updated). Sebastopol, CA: O'Reilly, 2007. ISBN 05-965-2722-5.

[49] Protocol-Independent Routing Properties Feature Guide. In: *Juniper* [on line]. 2019 [cit. 2019-12-09]. Dostupné z: https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration-statement/router-id-edit-routing- options.html

[50] TISO, John a Keith HUTTON. *Designing Cisco network service architectures (ARCH)*. 3rd ed. Indianapolis, IN: Cisco Press, 2012. ISBN 15-871-4288-0.

[51] Cisco Config Analysis Tool. *GitHub* [online]. [cit. 2019-12-11]. Dostupné z: https://github.com/cisco-config-analysis-tool/ccat

[52] Router connectivity visualization and compliance auditing. *GitHub* [online]. [cit. 2019-12-11]. Dostupné z: https://github.com/asifhj/Router-Auditing-Tool

[53] SWAROOP, C H. A Byte of Python. In: *Swaroopch* [online]. [cit. 2019-12-14]. Dostupné z: https://python.swaroopch.com/about_python.html

[54] YAML: YAML Ain't Markup Language. In: *YAML* [online]. [cit. 2019-12-14]. Dostupné z: https://yaml.org/

[55] Python RegEx. In: *W3schools* [online]. [cit. 2019-12-14]. Dostupné z: https://www.w3schools.com/python/python_regex.asp

Zoznam symbolov, veličín a skratiek

CIA	confidentiality, integrity, availability – dôvernosť, integrita, dostupnosť
DDoS	Distributed Denial of Service – distribuované odoprenie služby
DoS	Denial of Service – odoprenie služby
ACL	Access Control List – zoznam pre riadenie prístupu
CVSS	Common Vulnerability Scoring System
IDS	Intrusion Detection System – systém detekcie narušenia
IPS	Intrusion Prevention System – systém prevencie prienikov
FHRP	First Hop Redundancy Protocol
SNMP	Simple Network Management Protocol
AAA	Authentication Authorization Accounting
SSH	Secure Shel
OSPF	Open Shortest Path First
LAN	Local Area Network
IP	Internet Protocol
IPv6	Internet Protocol version 6
VLAN	Virtual LAN
ARP	Address Resolution Protocol
MAC	Media Access Control
LLDP	Link Layer Discovery Protocol
CDP	Cisco Discovery Protocol
API	Application Programming Interface
GUI	Graphical User Interface – grafické užívateľské rozhranie
uRPF	Unicast Reverse Path Forwarding
BGP	Border Gateway Protocol
TTL	Time To Live
HTTP	Hypertext Transfer Protocol
NTP	Network Time Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
PTP	Precision Time Protocol
SCP	Secure Copy Protocol
TFTP	Trivial File Transfer Protocol
SFTP	Secure File Transfer Protocol
ICMP	Internet Control Message Protocol
VPN	Virtual Private Network – Virtuálna privátna sieť

PPTP	Point-to-Point Tunneling Protocol
L2TP	Layer 2 Tunneling Protocol
IPSec	IP Security
MLD	Multicast Listener Discovery
IGMP	Internet Group Management Protocol
VRRP	Virtual Router Redundancy Protocol
HSRP	Hot Standby Redundancy Protocol
GLBP	Gateway Load Balancing Protocol
STP	Spanning Tree Protocol
DTP	Dynamic Trunking Protocol
EAP	Extensible Authentication Protocol
PEAP	Protected Extensible Authentication Protocol
EAPoL	Extensible Authentication Protocol over LAN
VTP	Virtual Trunking Protocol
MVRP	Multiple VLAN Registration Protocol
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
SEND	Secure Neighbor Discovery

Zoznam príloh